

Cognitive Science Tutorial

APEX

An Architecture for Modeling Human Performance in Applied HCI Domains

Michael Dalal (mdalal@arc.nasa.gov)

Michael Freed (mfreed@arc.nasa.gov)

Robert Harris (rharris@arc.nasa.gov)

Michael Matessa (mmatessa@arc.nasa.gov)

John Rehling (jrehling@arc.nasa.gov)

Roger Remington (rremington@arc.nasa.gov)

Alonso Vera (avera@arc.nasa.gov)

NASA Ames Research Center

1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



Day Plan

9:00 - 9:30	A Brief Introduction to Apex	2:00 - 3:30	Exercise #3: CPM - GOMS
9:30 -10:30	Learning to model in Apex	3:30 - 3:34	<i>Break</i>
10:30 - 10:45	<i>Break</i>	3:45 - 4:45	Exploring Apex
			Calculator world
			Template Building
			Vision demonstration
			Website
			Sherpa exploration
11:30 - 12:30	Exercise #2: KLM	4:45 - 5:00	Wrap-Up
12:30 - 2:00	<i>Lunch</i>		



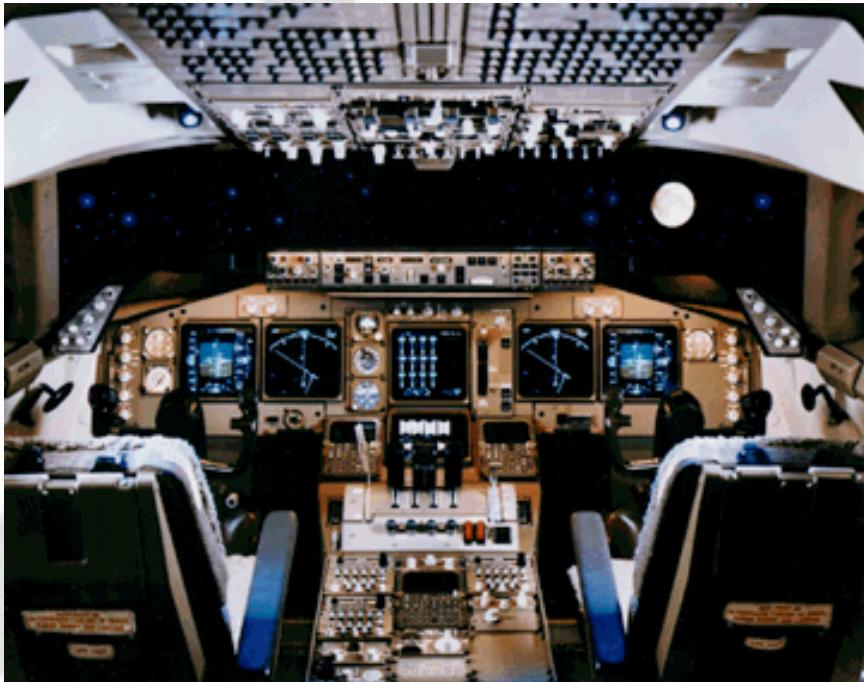
What we're going to show you

- How to build GOMS models in Apex
- How Apex automatically handles the resource assignments underlying CPM-GOMS
 - 2 mouse movement templates from Gray & Boehm-Davis (2000)
 - Rules for combining them for maximum parallelism
- How such templates could be used to build other related model
 - Community of template builders contributing to a common pool

A Brief Introduction to Apex

- Motivation and Goals
- Apex Architecture

Complex dynamic environments



- Users must make timely inputs
- Users juggle multiple tasks
- Predictability: Similar patterns occur over time
- Unpredictability: Interruptions occur

Motivations

- Modeling system that can make useful predictions about skilled operator behavior in complex dynamic environments
- Make cognitive modeling more accessible to non-specialists, especially in the design phase

Resource Allocation

- A multi-tasking agent must allocate resources proactively and reactively
 - Mechanisms for task suspension and recovery
 - Mechanisms for parallel task execution subject to *resource constraints* and *logical dependencies*
- Apex is built around a *reactive planner*
 - Designed for high-level task juggling
 - Also handles the interleaving of low-level resource demands needed for HCI modeling



HCI and GOMS Modeling

- Interleaving of resources is also critical for GOMS models that make useful HCI predictions
- The modeling process is:
 - labor intensive
 - error prone
 - requires specialized knowledge of human cognition and modeling methodologies
 - takes engineers out of their domain of expertise
- Modeling will not become a common tool for engineering development until these obstacles are overcome

Usability Issues with Computational Cognitive Modeling

- Problem
 - Human performance data not in a form easily accessible to non-specialists
 - Model building time-consuming, complex, error prone (even for experts)
- Approach
 - Reusable templates that incorporate human performance characteristics
 - Software tools that integrate templates into large-scale models



GOMS in Apex

- We have applied Apex to a simple HCI example
 - Use predefined CPM-GOMS templates to predict performance
- Reactive resource allocation in Apex handles low-level resource competition
 - allows behavior to emerge from a sequence of pre-defined templates
 - automatically interleave templates
 - supports hierarchical task decomposition that simplifies CPM-GOMS modeling

Reusable Templates

- Package the abundance of data on human perceptual, cognitive, and motor phenomena into a set of modules (templates) that can be directly incorporated into models
- Templates reduce the amount of psychology and modeling methodology required to build models
 - Compile the human performance data into templates
 - Focus the modeler on task analysis
 - Computer-supported usability analysis

Some Existing Templates

- Cognitive modeling has made wide use of general performance characteristics
 - Fitts's Law
 - Vision
 - Short-term memory size
 - Speaking and reading rates
- Some have been incorporated into larger modules
 - Keystroke templates (John & Gray, 1992; Gray, John, & Atwood, 1993)
 - Auditory & verbal templates (John & Gray, 1992; Gray, et al., 1993)
 - Conversation (John & Gray, 1992; Gray, et al., 1993)
 - Transcript typing (John, 1996)
 - Mouse move-and-click routines (Gray & Boehm-Davis, 2000)



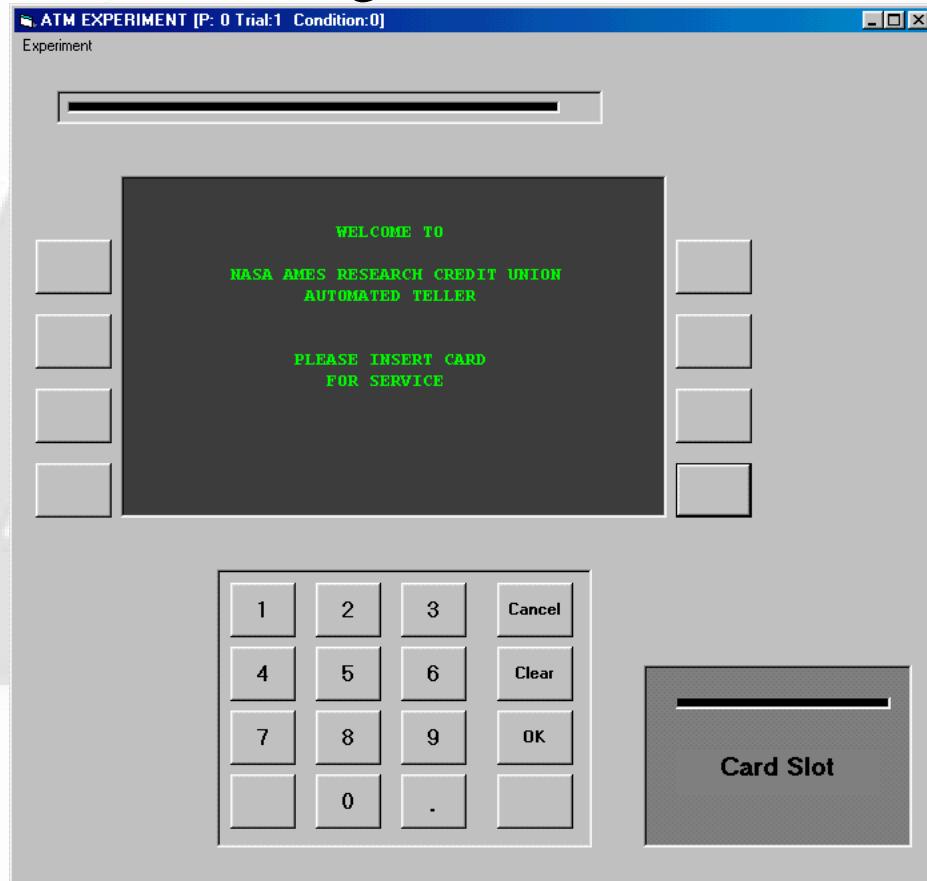
Text Legibility Example

1 August 2001

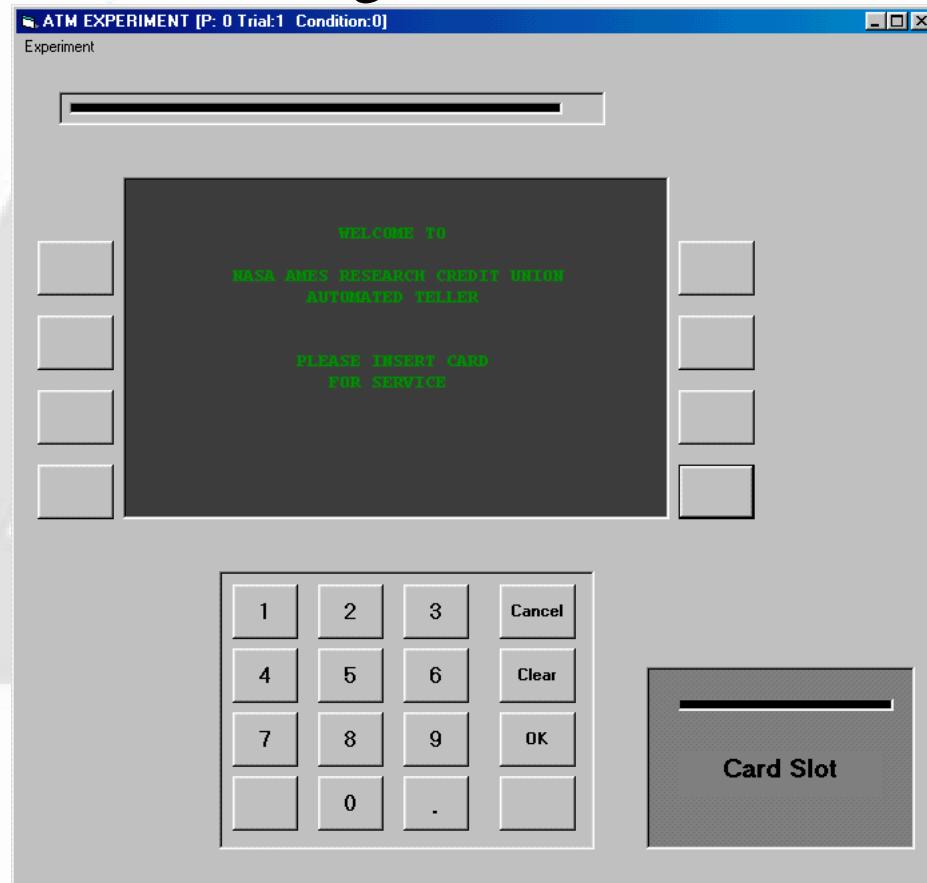
Cognitive Science 2001
Edinburgh, Scotland



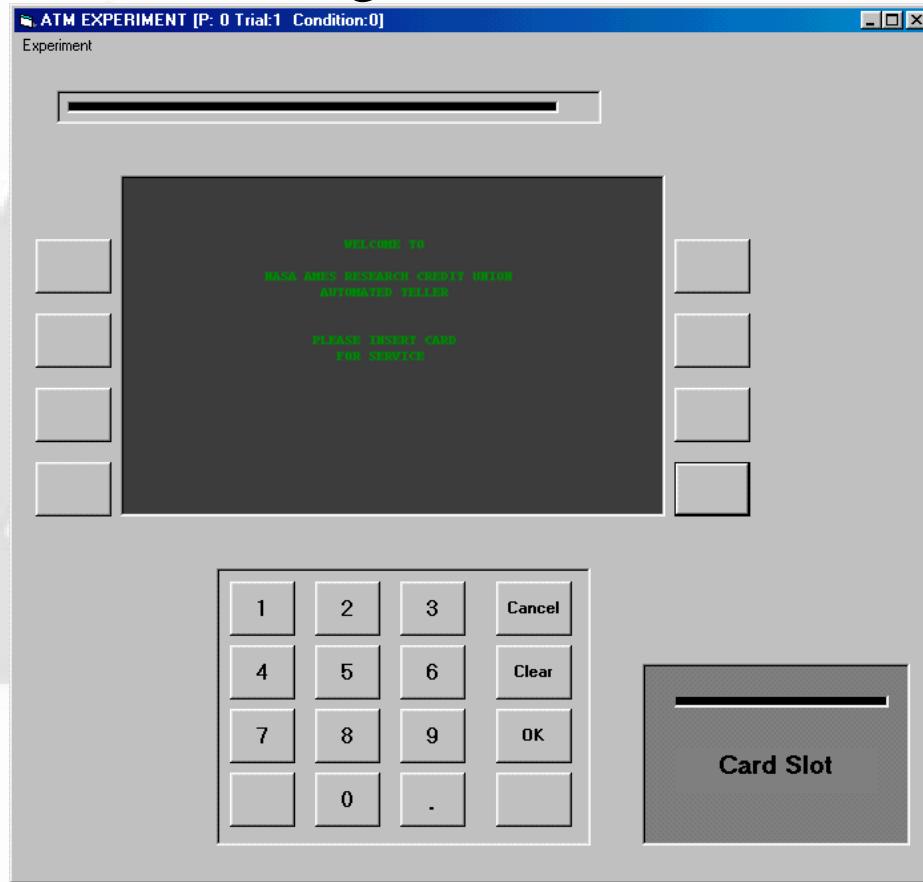
Layout #1: lime (0 255 0) on (46 46 46); letter size 0.28°
RT of reading: 3349 ms



Layout #2: green (0 128 0) on (46 46 46); letter size 0.28°
RT of reading: 3463 ms



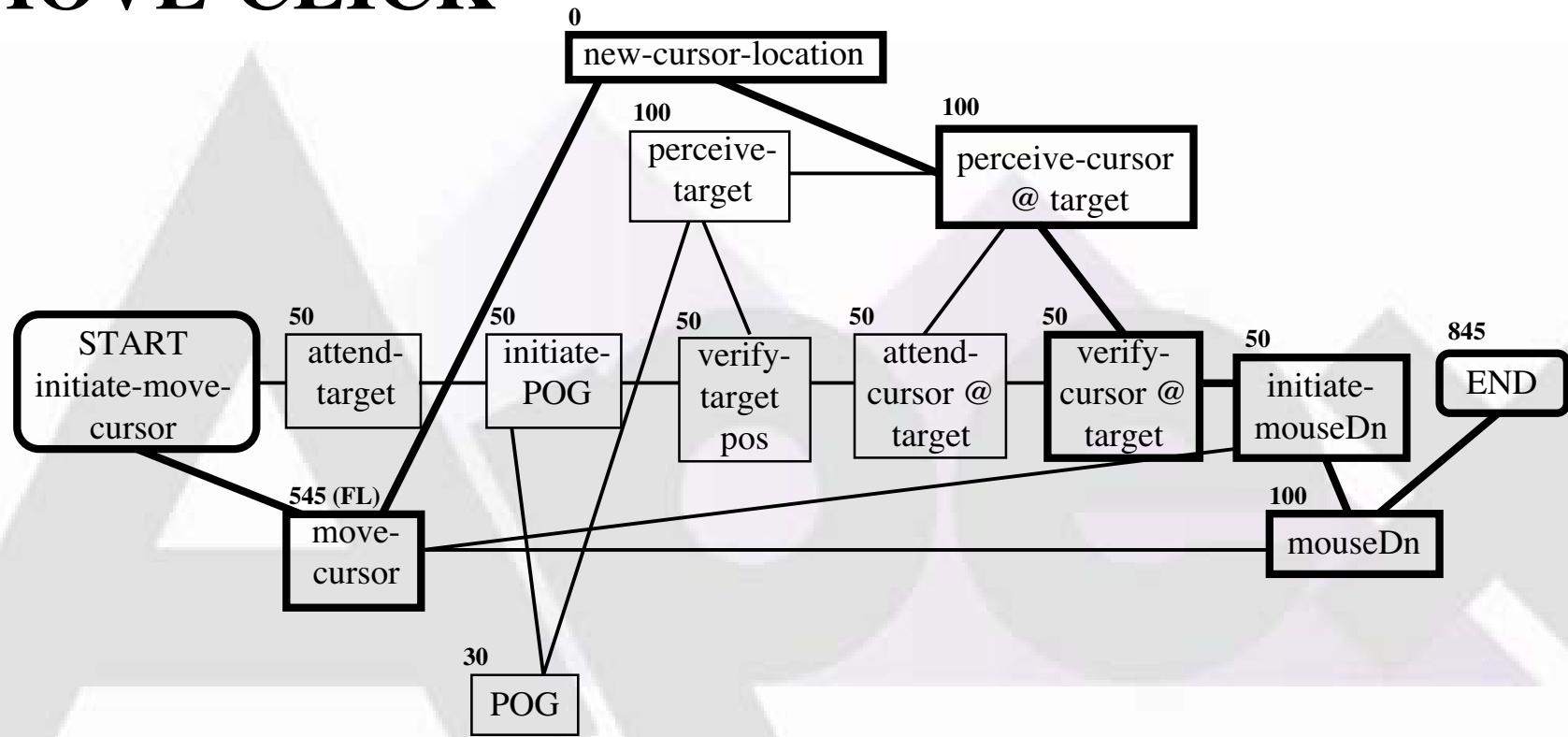
Layout #3: green (0 128 0) on (46 46 46); letter size 0.21°
RT of reading: 3977 ms



Mousing Templates

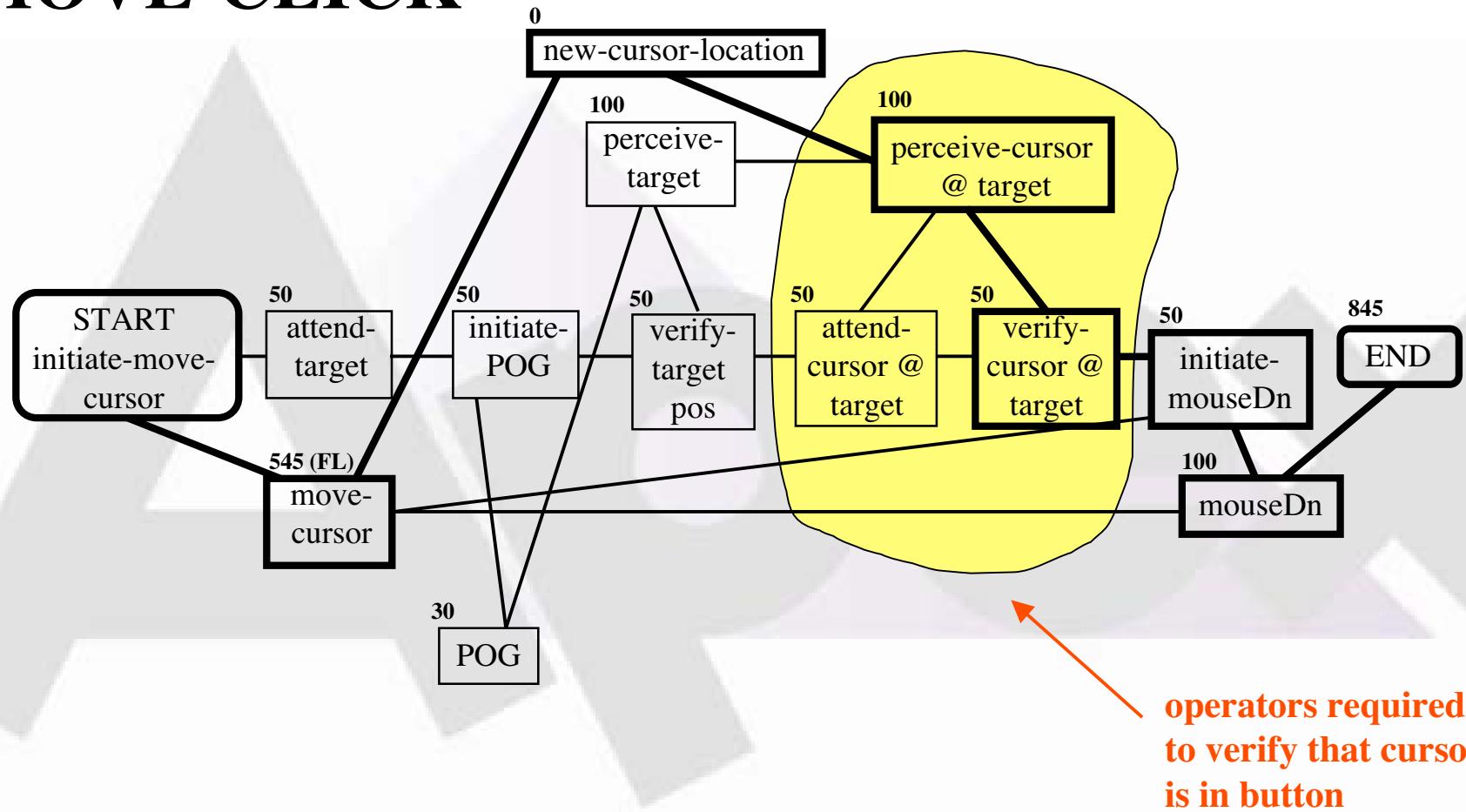
- Gray & Boehm-Davis specify distinct micro-strategies that arise with practice in repetitive perceptual-motor tasks.
 - slow mouse and click
 - fast mouse and click
 - slow click and mouse
 - medium click and mouse

SLOW MOVE-CLICK

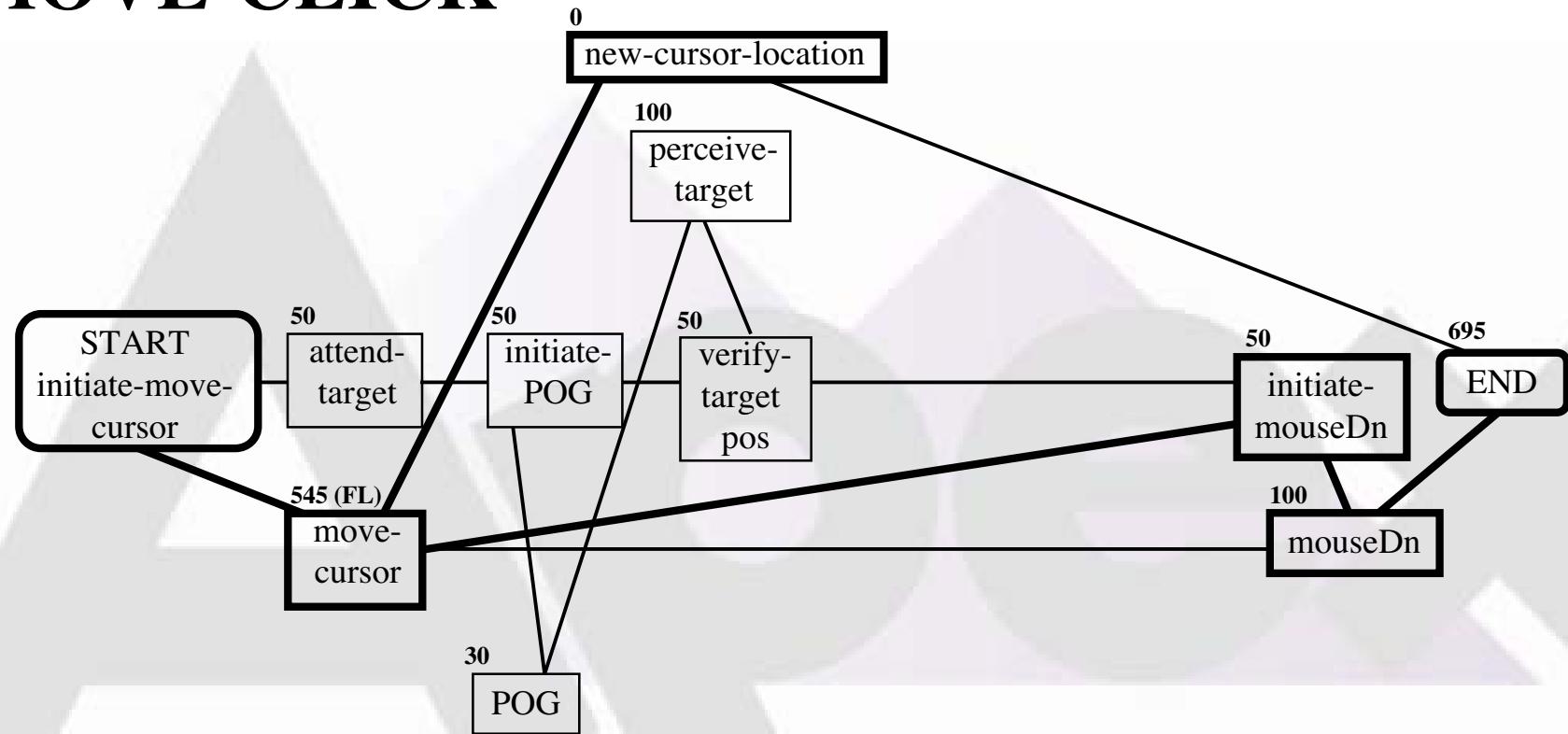


- from *Gray and Boehm-Davis (2000)*

SLOW MOVE-CLICK



FAST MOVE-CLICK



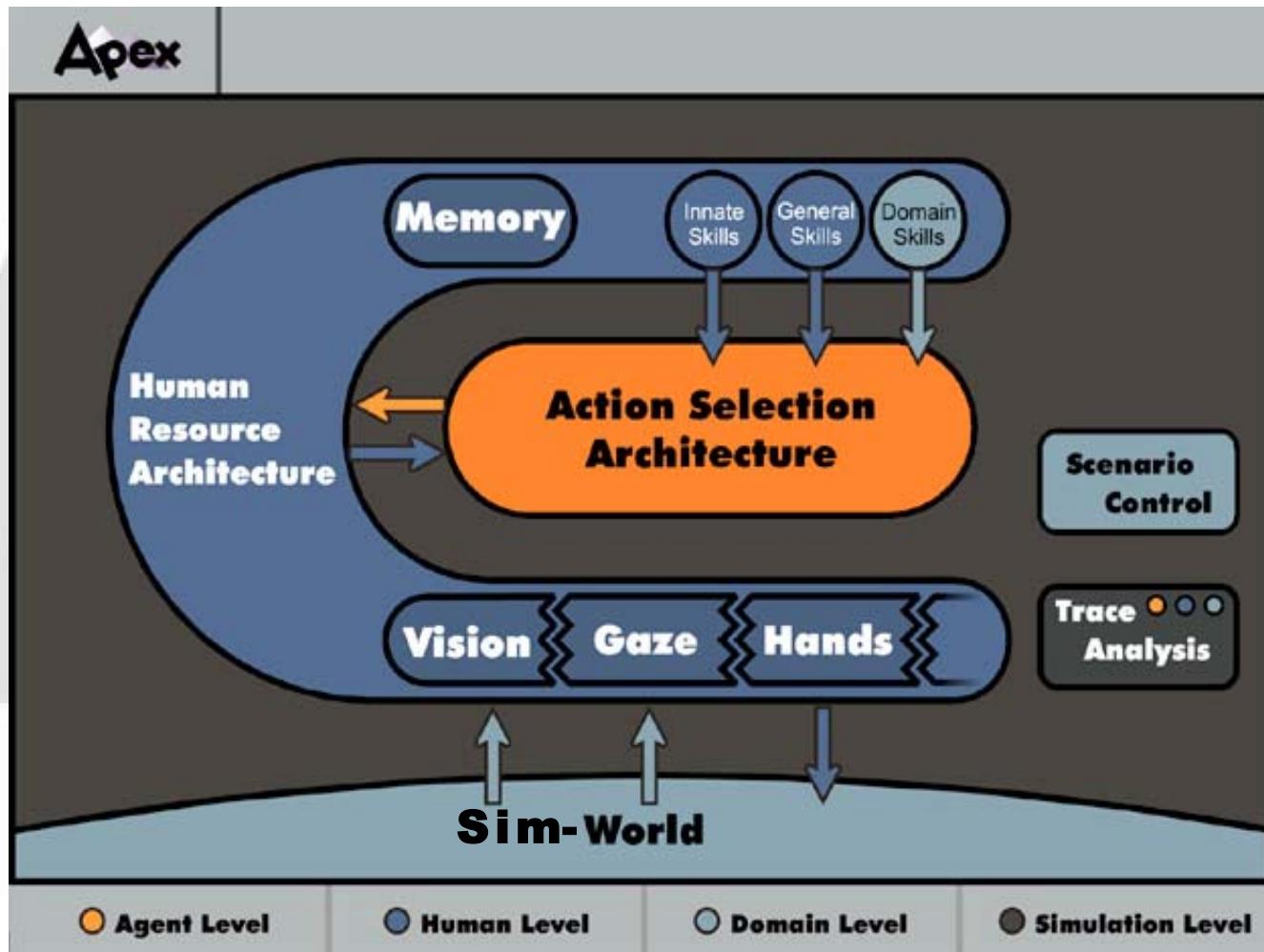
- from *Gray and
Boehm-Davis (2000)*

Tutorial Goals

- To introduce you to Apex and get a sense of what it can do and how it works
 - Build GOMS models a simple ATM interface task
 - Learn how to program in Apex
- To discuss Reusable Behavior Templates, and to use a simple interface modeling task to demonstrate how Apex can support GOMS modeling
- Excite you about building reusable templates that would extend Apex capability
 - Number of human behaviors of interest is very large
 - Only a persistent community effort will make headway



The Apex Architecture



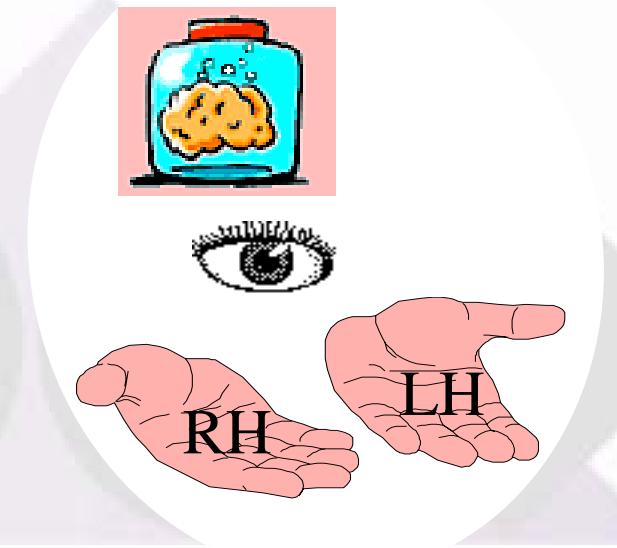
1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



The Human Resource Architecture

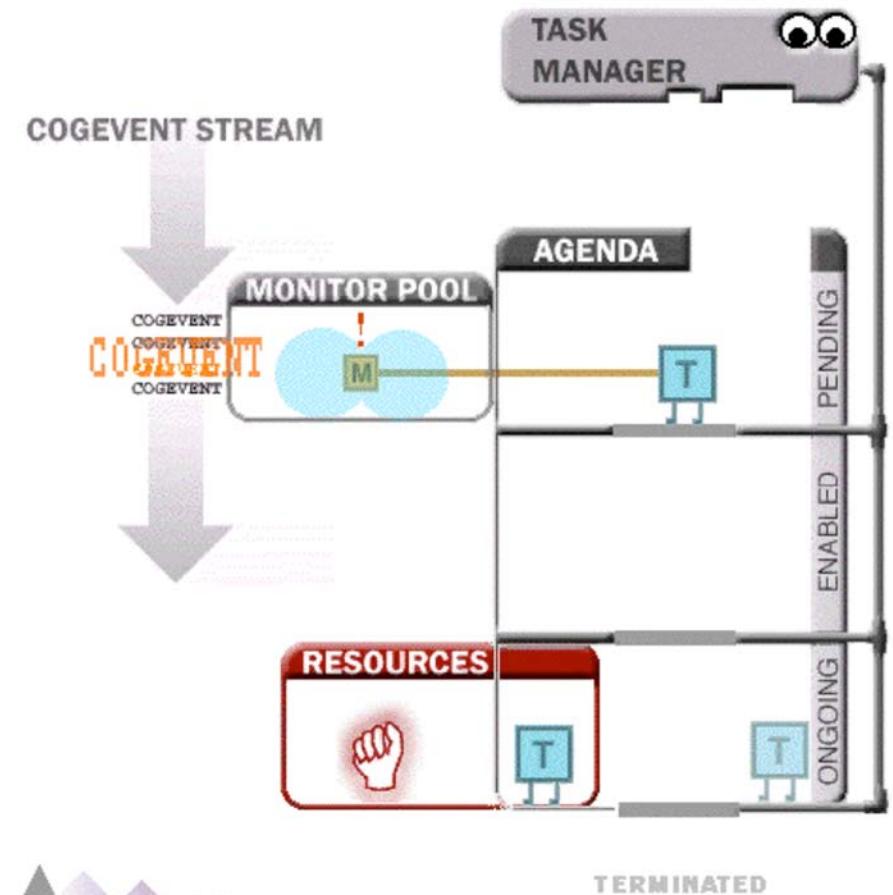
- Simulates the output of perceptual, cognitive, and motor processing
- Captures processing times and resource constraints
- Supports interruption from
 - external events
 - internal events



- cognition
- perception (mostly vision)
- motor

The Action Selection Architecture

- Implements a reactive planner that supports
 - Sketchy plans
 - Hierarchical task decomposition
 - Multitasking
- Maximizes parallel processing subject to
 - Resource constraints
 - Data dependencies
- Resource conflicts settled by task priority



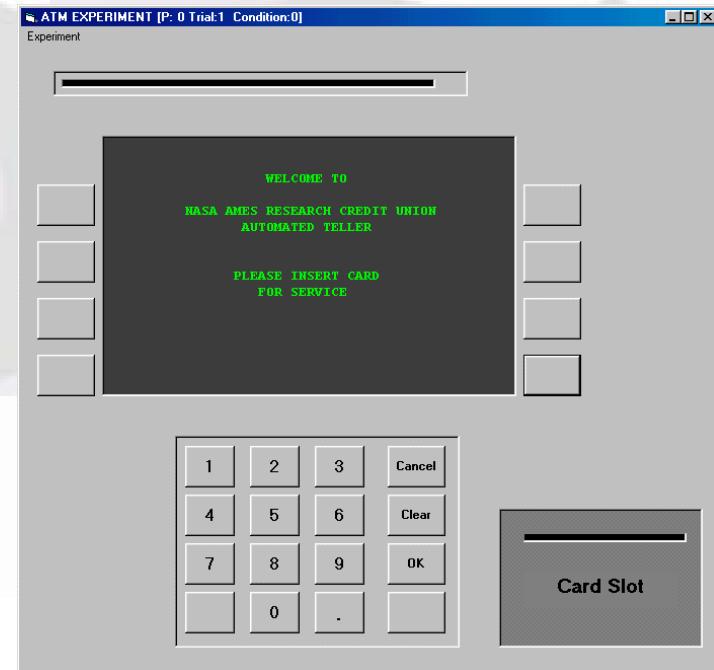
Apex

Basic PDL Concepts

- **procedure:** instructions, like a recipe, for accomplishing a given goal
- **step:** a separable and distinct “thing to do” as a part of larger procedure.
- **waitfor:** represents sequencing information in the instructions or recipe. What needs to have happened already, for the next step to be enabled.

A Simple Task Model

- get money from an ATM
- start with a goal-based hierarchical decomposition
- a coarse model



Hierarchical Task Decomposition in PDL

```
(procedure
  (index (do banking))
  (step s1 (initiate session))
  (step s2 (do transaction) (waitfor ?s1))
  (step s3 (end session) (waitfor ?s2))
  (step s4 (terminate) (waitfor ?s3)))

(procedure
  (index (initiate session))
  (step s1 (insert card))
  (step s2 (enter password) (waitfor ?s1))
  (step s3 (terminate) (waitfor ?s2)))
```

CMN-GOMS (incomplete)

- do banking
 - initiate session
 - insert card
 - enter password
 - do transaction
 - end session

PDL decomposition of `initiate session`

```
(procedure
  (index (insert card))
  (profile memory)
  (step s1 (start-activity memory memory-act :duration 1021 => ?a))
  (step s2 (terminate) (waitfor (completed ?a)))))

(procedure
  (index (enter password))
  (profile memory)
  (step s1 (start-activity memory memory-act :duration 2428 => ?a))
  (step s2 (terminate) (waitfor (completed ?a))))
```

durations
from data

Modeling in Apex

1 August 2001

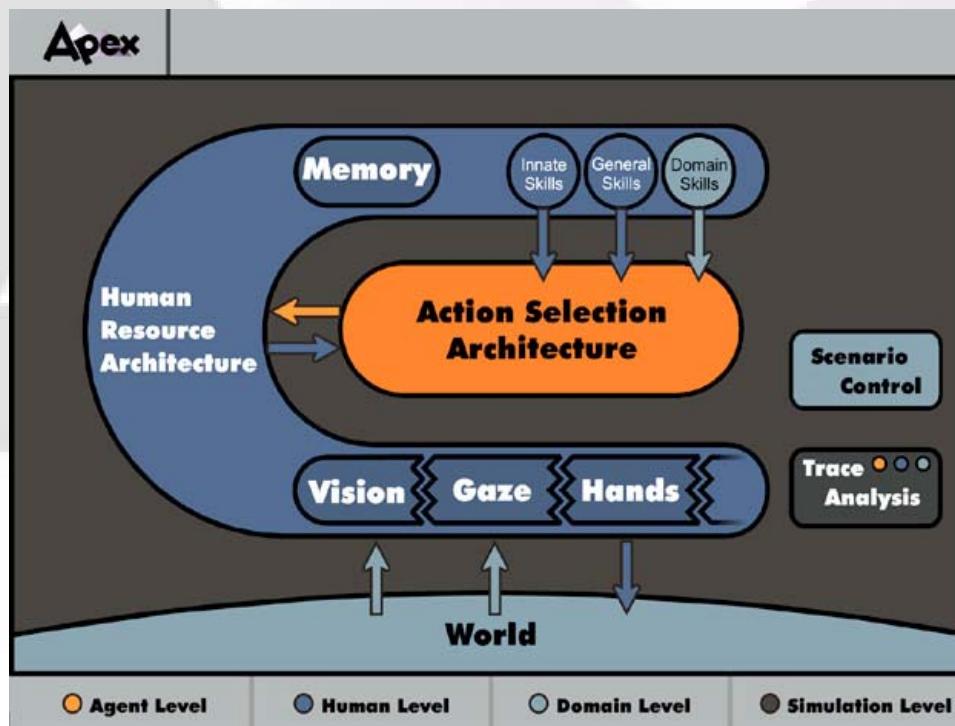
Cognitive Science 2001
Edinburgh, Scotland



Section Overview

Modeling in Apex

GOAL: start modeling in 45 minutes



1. Components of an Apex model
2. How the ASA works
3. Writing PDL
4. A scripted exercise

Components of an Apex model

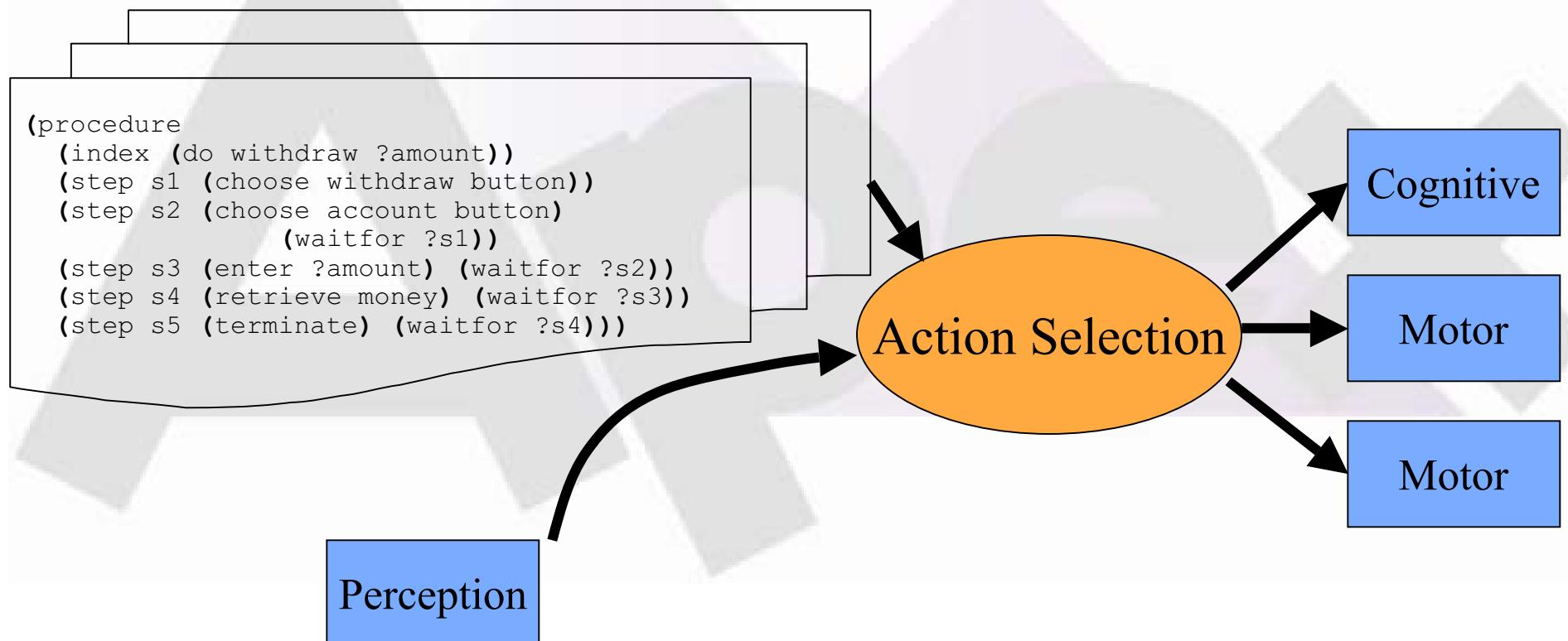
An Apex model (simworld) consists of:

1. PDL procedures
2. Object definitions (e.g. ATM)
3. Scenario specifications
 - objects to include in a simulation trial
 - object initial-state: location, color, on/off, initial goals
 - object relations: components, relative position, attachment...
 - humans specified uniformly with other physical objects
 - resources (e.g left-hand, vision) are components
 - OK to alter default configuration (e.g. no right hand)



The Action Selection Architecture

Function: to turn procedures + perception into behavior



ASA Animated View

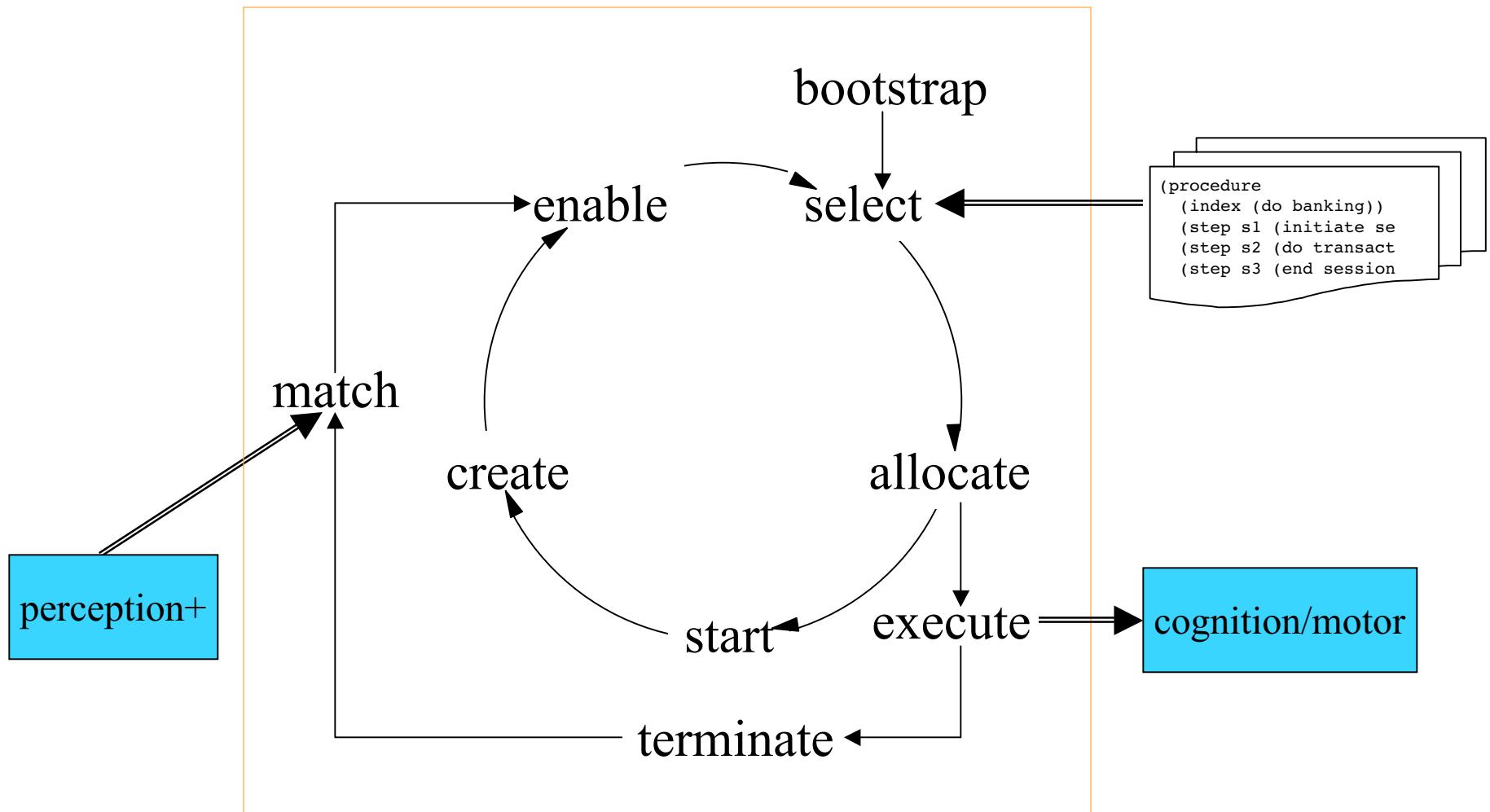
1 August 2001

Cognitive Science 2001
Edinburgh, Scotland

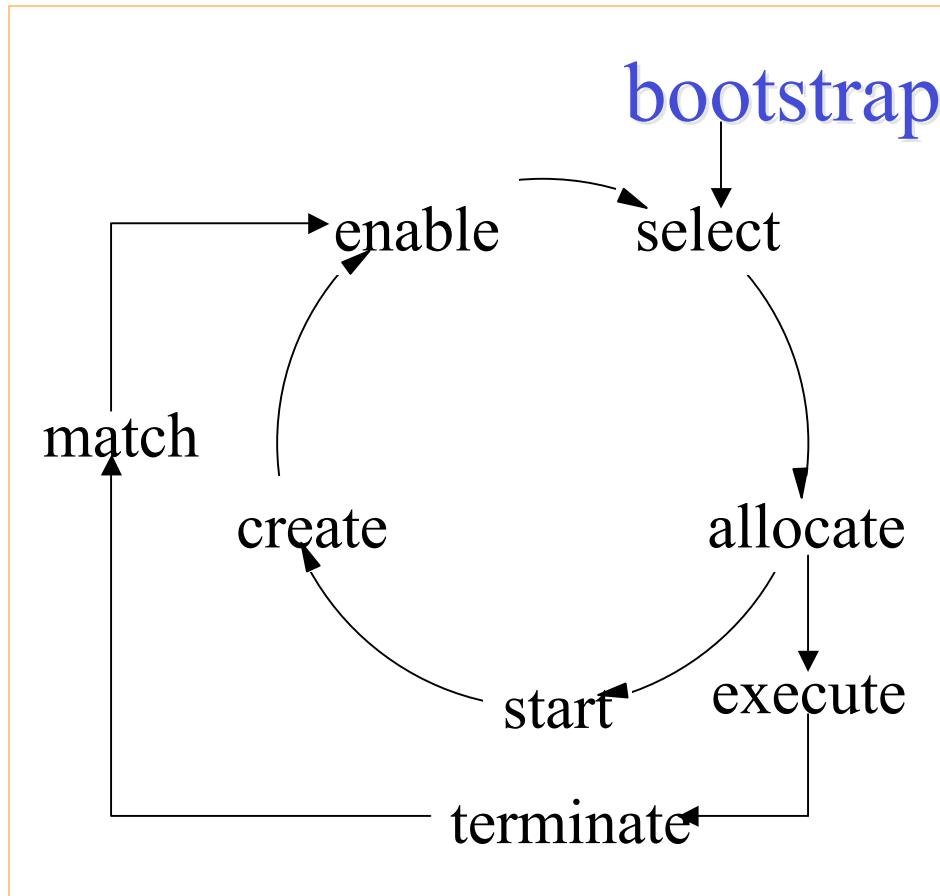


Action Selection Architecture

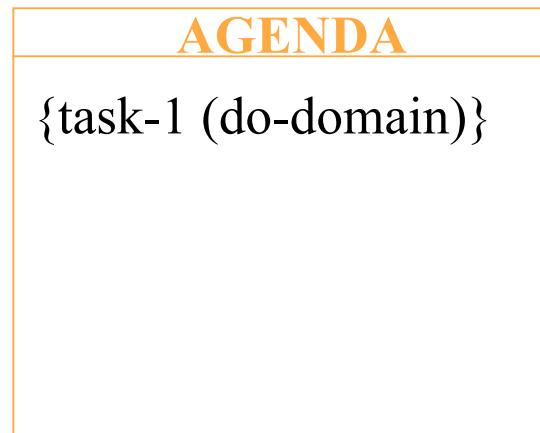
The basic task execution cycle



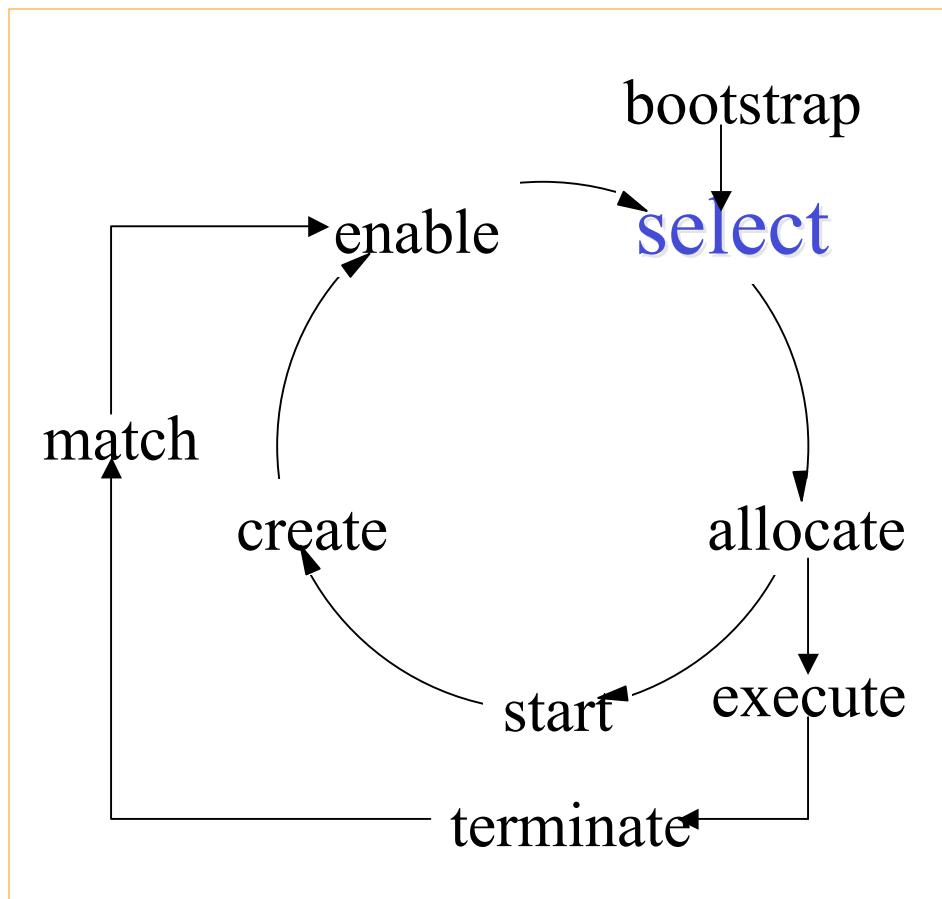
Bootstrapping task execution



- Initialization **creates**:
 $\{task-1\ (do\text{-}domain)\}$
- Tasks are stored on the **agenda**.



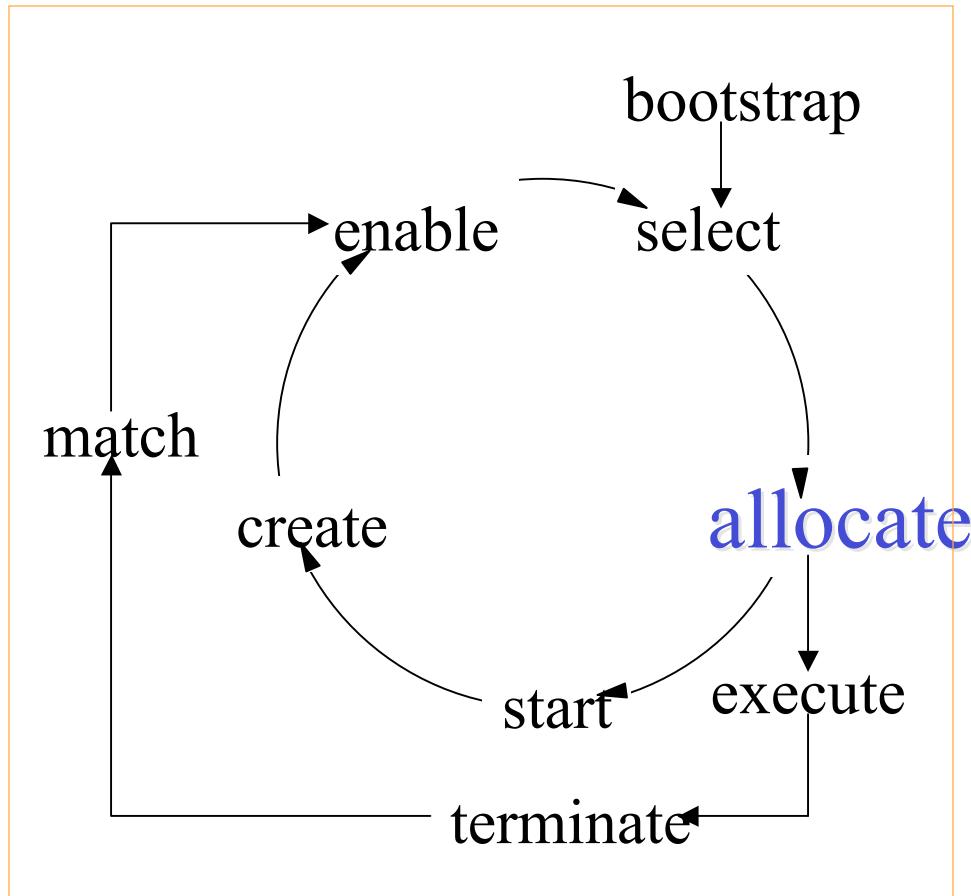
Procedure Selection



- Selection **matches** task to index clause of stored procedure
- Modeler specifies a do-domain procedure

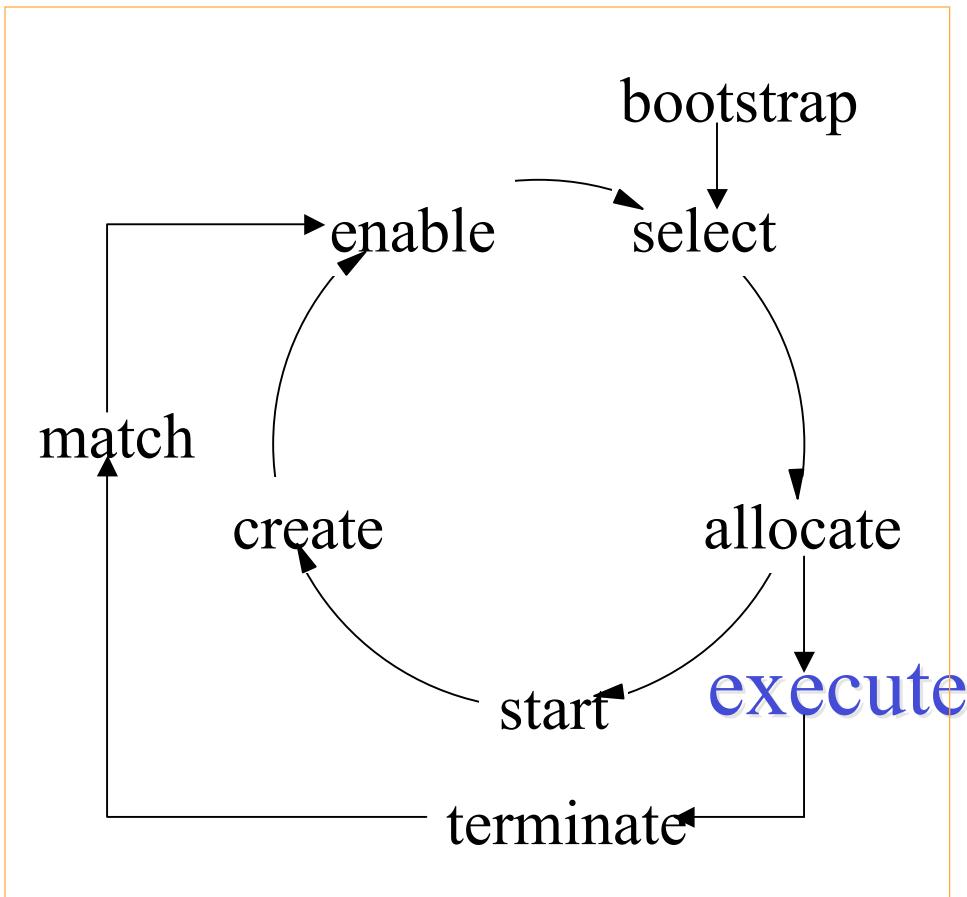
```
(procedure  
  (index (do-domain))  
  (step s1 (walk to ATM))  
  (step s2 (withdraw 80 from ATM)  
           (waitfor ?s1))  
  (step s3 (walk away from ATM)  
           (waitfor ?s2))  
  (step s4 (terminate)  
           (waitfor ?s3))))
```

Resource Allocation



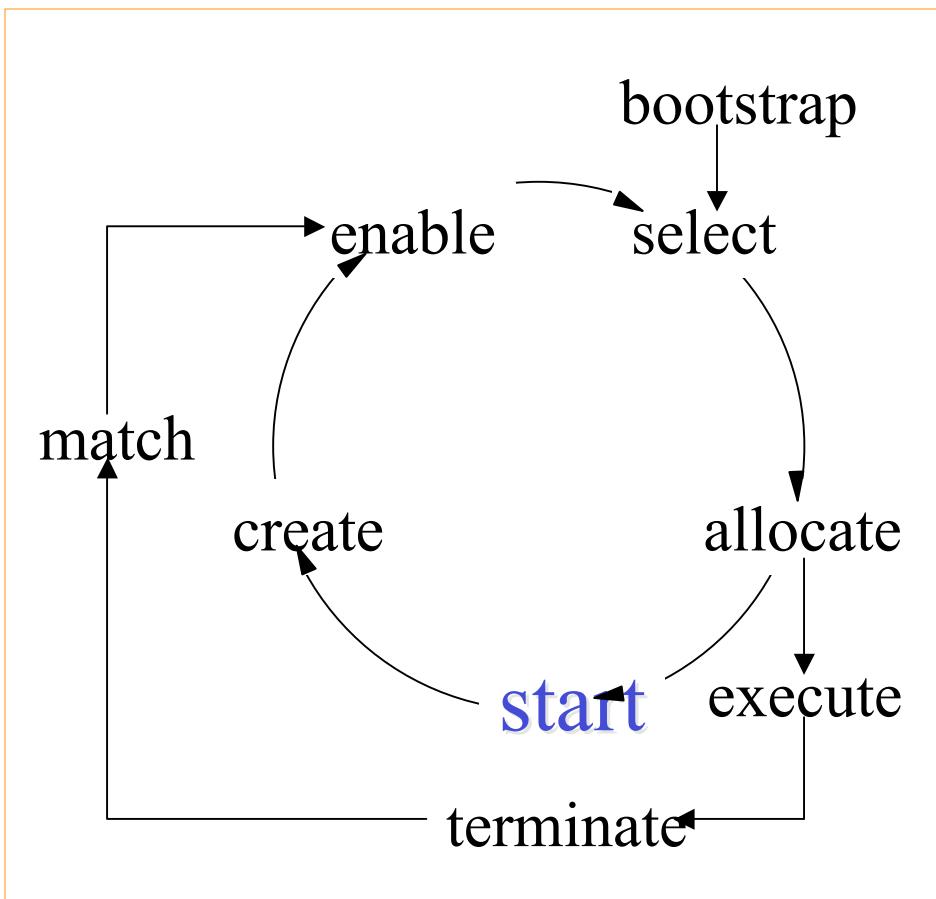
- Procedures can specify resource requirements
- If only one task needs the resource, it is allocated to that task. Otherwise a conflict is detected and resolved

Primitive Task Execution



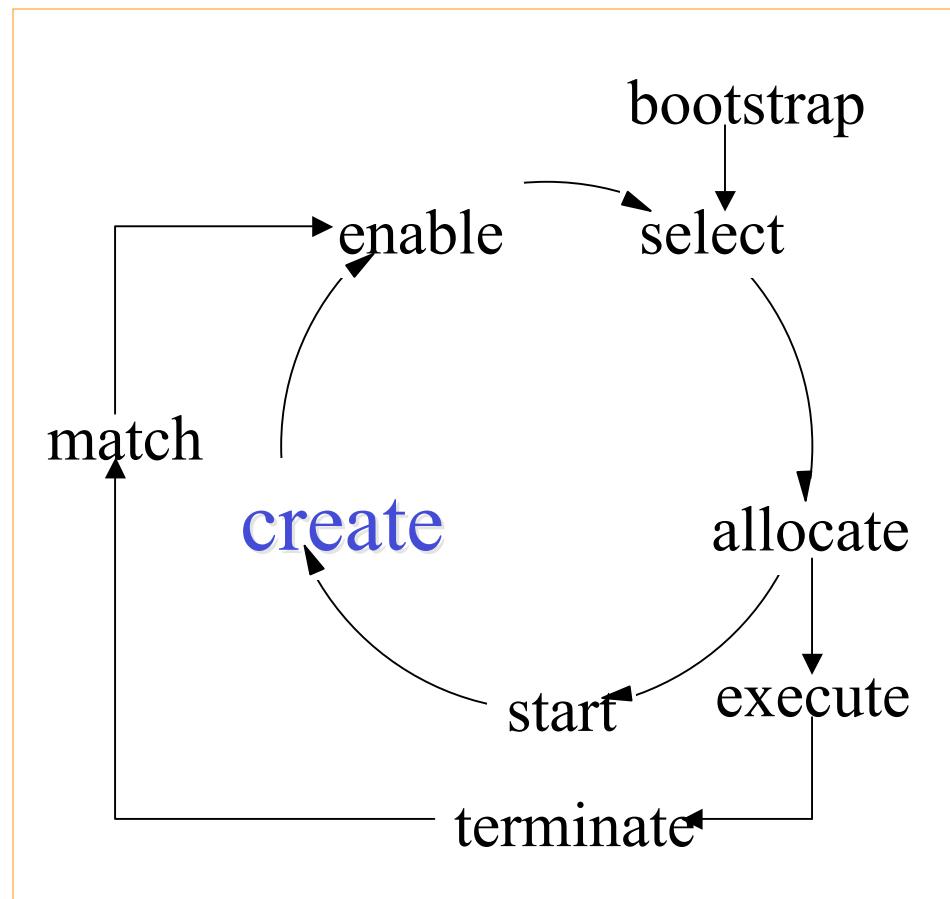
- Primitive tasks can be executed immediately
- One kind of primitive, **start-activity**, begins a resource action
- **Terminate** primitives cause some other task to end

Starting a Task



- When a non-primitive task is allocated needed resources, it is started
- Starting changes it from **enabled** to **ongoing**
- Other possible task states: **pending** and **terminated**

Creating Tasks and Monitors

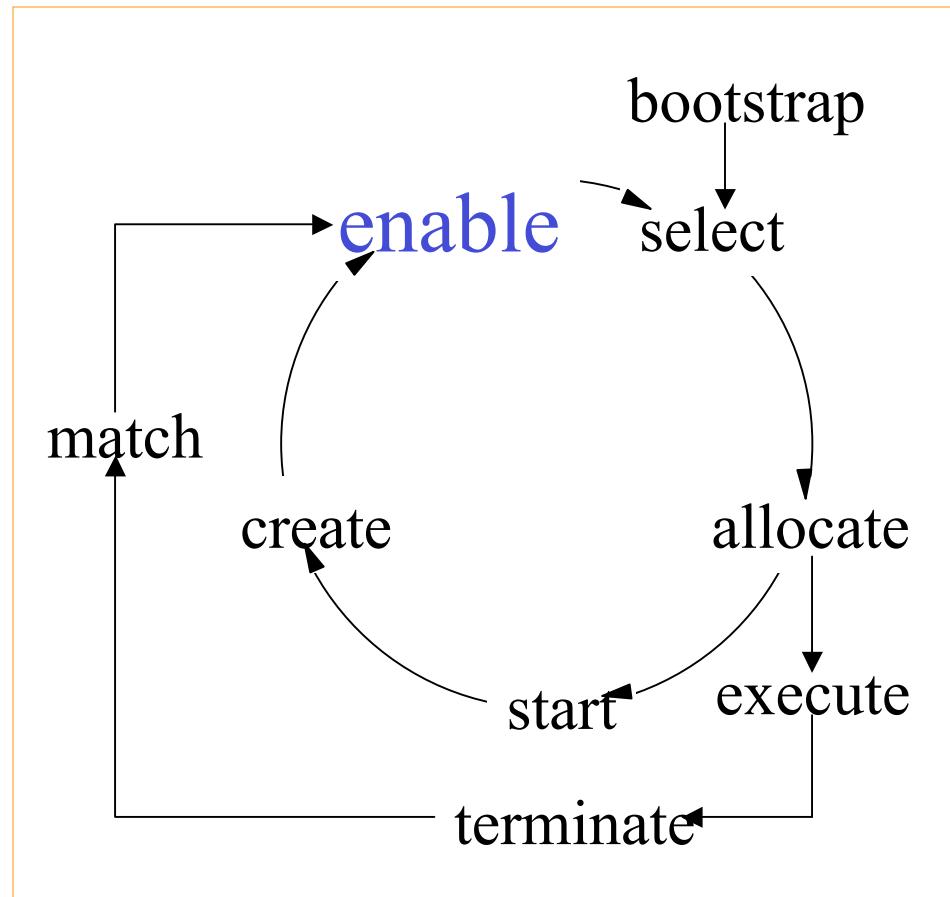


- Starting a task creates one new task each step in the selected procedure
- a **monitor** is created to look out for every waitfor condition

AGENDA

{task-1 (do-domain)}
{task-2 (walk to ATM)}
{task-3 (withdraw 80...)}
 {monitor-1 (*terminated task-2*)}
{task-4 (walk away ...)}
{task-5 (terminate task-1)}

Task Enablement

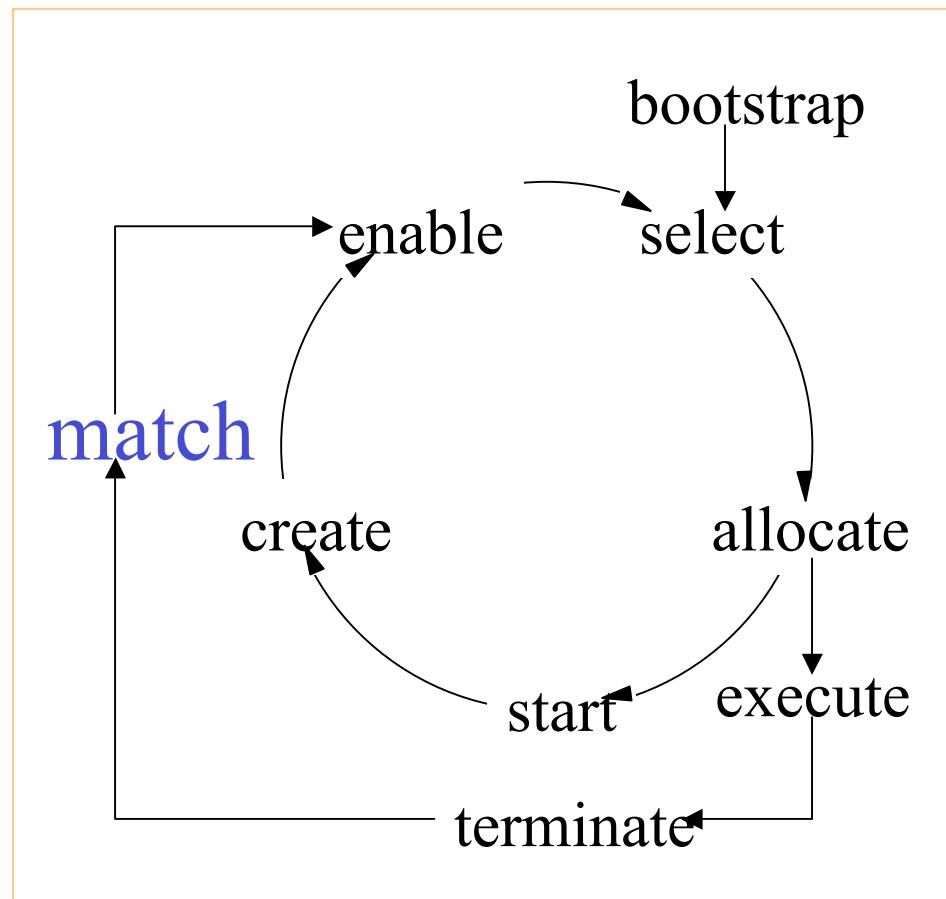


- A task with no associated waitfor preconditions starts in the **enabled** state
- Else, it starts **pending** until all preconditions satisfied

AGENDA

{task-1 (do-domain)}
{task-2 (walk to ATM)}
{task-3 (withdraw 80...)}
{monitor-1 (terminated task-2)}
{task-4 (walk away ...)}
{task-5 (terminate task-1)}

Monitors and Cogevents



- Perceptual resources make **cogevents** representing perceived changes
e.g. *(color {obj-5} green)*
- The ASA makes cogevents related to task execution
e.g. *(terminated {task-2})*
- A monitor is **satisfied** when it matches a cogevent. A task is enabled when all its monitors are satisfied

Goals of PDL

- Expressive, easy-to-learn, formal language
- Represent how-to knowledge
- Model behavior in dynamic, uncertain environments

PDL Syntax: Parentheses

```
(procedure
  (index (do withdraw ?amount))
  (step s1 (choose withdraw button))
  (step s2 (choose account button) (waitfor ?s1))
  (step s3 (enter ?amount) (waitfor ?s2))
  (step s4 (retrieve money) (waitfor ?s3))
  (step s5 (terminate) (waitfor ?s4))))
```

- PDL is based on Lisp and uses parentheses to enclose clauses
 - **Procedure-level clauses** (e.g., index, step)
 - **Step-level clauses** (e.g., waitfor)

The PROCEDURE Clause

(**procedure**

```
(index (do withdraw ?amount))  
(step s1 (choose withdraw button))  
(step s2 (choose account button) (waitfor ?s1))  
(step s3 (enter ?amount) (waitfor ?s2))  
(step s4 (retrieve money) (waitfor ?s3))  
(step s5 (terminate) (waitfor ?s4)))
```

- By default, steps in a procedure specify concurrent tasks
- Task order can be imposed by **waitfor** clauses or by resource-conflicts

The INDEX clause

```
(procedure
  (index (do withdraw ?amount) )
  (step s1 (choose withdraw button))
  (step s2 (choose account button) (waitfor ?s1))
  (step s3 (enter ?amount) (waitfor ?s2))
  (step s4 (retrieve money) (waitfor ?s3))
  (step s5 (terminate) (waitfor ?s4))))
```

- Informally, the **index** clause describes what goals the procedure can be used to achieve
- If the index matches an enabled task, the procedure is a valid way to break that tasks into subtasks
- Variables in an index become bound to values when a pattern match occurs



The STEP clause

```
(procedure
  (index (do withdraw ?amount))
  (step s1 (choose withdraw button))
  (step s2 (choose account button) (waitfor ?s1))
  (step s3 (enter ?amount) (waitfor ?s2))
  (step s4 (retrieve money) (waitfor ?s3))
  (step s5 (terminate) (waitfor ?s4))))
```

- Step clauses specify how to carry out the procedure
- The step-tag (e.g. s1) allows one step to refer to another
- The step's activity description may contain variables
- Waitfors and other step-level clauses may be added



The WAITFOR Clause

```
(procedure
  (index (do withdraw ?amount))
  (step s1 (choose withdraw button))
  (step s2 (choose account button) (waitfor ?s1))
  (step s3 (enter ?amount) (waitfor ?s2))
  (step s4 (retrieve money) (waitfor ?s3))
  (step s5 (terminate) (waitfor ?s4))))
```

- Waitfors determine when a step should be executed
- Several preconditions may be specified
- A precondition can be the **termination** of another step, perceptual event,...

Primitive action types

(procedure

```
(index (do withdraw ?amount))  
(step s1 (choose withdraw button))  
(step s2 (choose account button) (waitfor ?s1))  
(step s3 (enter ?amount) (waitfor ?s2))  
(step s4 (retrieve money) (waitfor ?s3))  
(step s5 (terminate) (waitfor ?s4)))
```

- Primitive tasks can be carried out immediately without being broken down into subtasks
- **Terminate** ends the task which invoked the procedure, e.g. *{task-11 (do withdraw 80)}*, and any unterminated descendants of that task



Interacting with resources

```
(procedure
  (index (choose ?button-type button))
  (profile right-hand)
  (step s1 (start-activity right-hand press :duration 1021 => ?act))
  (step s2 (terminate) (waitfor (completed ?act))))
```

- The PROFILE clause declares resource requirements
- Start-activity causes a resource (e.g. right-hand) to begin an activity (like a task but in a resource, not in the ASA)
- The *:duration* parameter can be a fixed value or a function such as #'*compute-fitts*
- It is possible to wait for the **completion** of an activity

The Priority Clause

```
(procedure
  (index (do withdraw ?amount))
  (step s1 (choose withdraw button) (priority 5))
  (step s2 (choose account button) (priority 3)))
  (step s3 (enter ?amount) (waitfor ?s2))
  (step s4 (retrieve money) (waitfor ?s3))
  (step s5 (terminate) (waitfor ?s4)))
```

- When two tasks need the same resource to execute, the ASA detects a **resource conflict**
- Conflicts are resolved in favor of highest priority task
- Tasks that do not conflict can be executed in parallel



PDL Functionality

task logic
resource constraints
temporal constraints
statistical regularities
utility
policies
contingency handling
background tasks
rationale



Covered in today's tutorial

Topics for a future tutorial

Practicalities

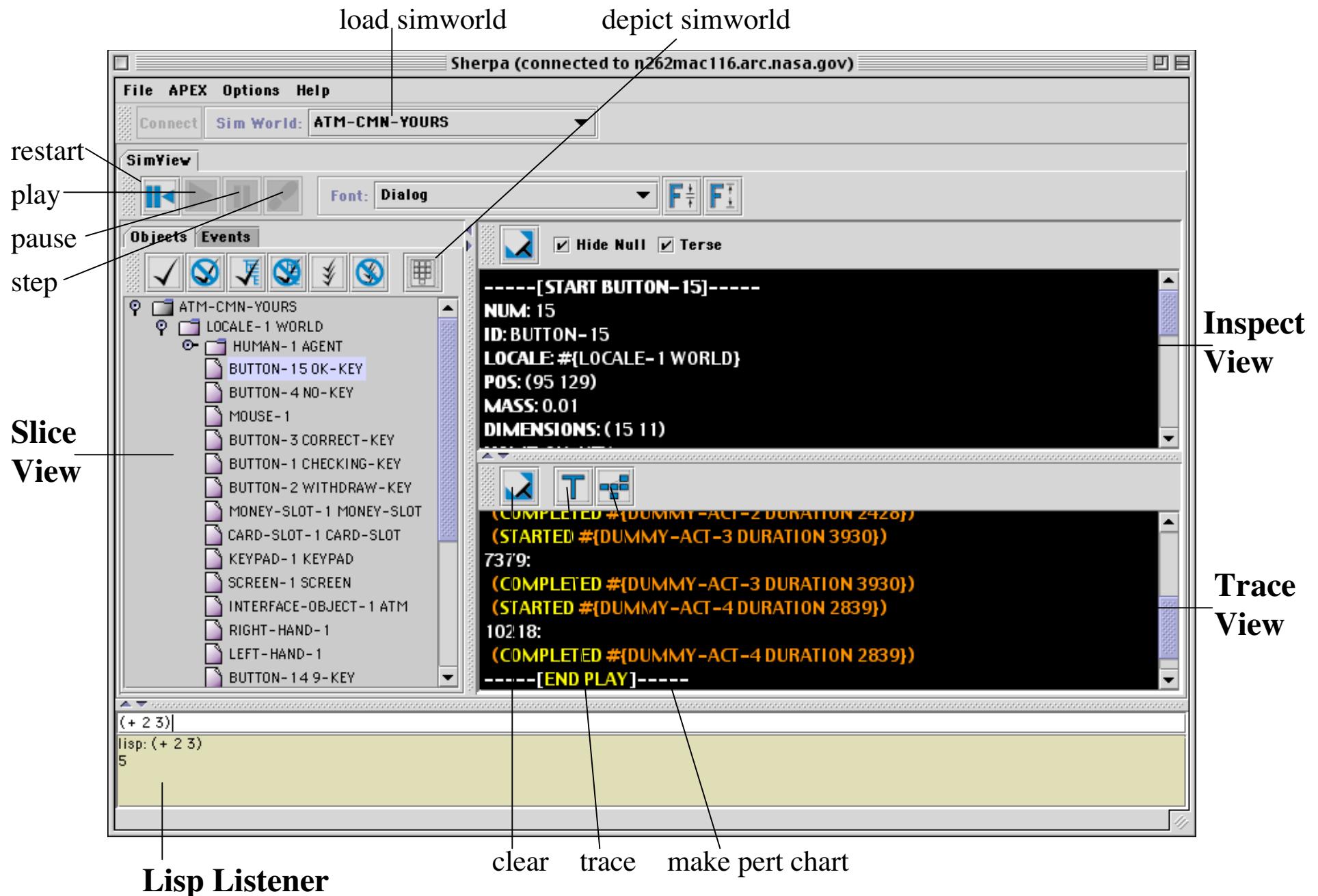
The Apex Modeling environment

- Macintosh Common Lisp editor
creating/viewing/modifying PDL
- Sherpa
running a simulation and analyzing results

Hands-on Exercise 1

System Overview

- load and run a model using Sherpa:
 - start, pause, restart
- move between major Sherpa views:
 - runtime, PDL, scenario, objects
- functions of the runtime viewer:
 - trace control, object inspection, ...
- view a PDL procedure
- view timeline



ATM Modeling

2:00 - 4:15 pm

1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



ATM Modeling

- Interactive Blackboard Exercise:
 - write CMN-GOMS model of an ATM task
- Interactive Programming Exercise:
 - Implement CMN-GOMS model in PDL
- Interactive Programming Exercise:
 - Transform CMN-GOMS model in PDL into a KLM model
- Interactive Programming Exercise:
 - Transform CMN-GOMS into CPM-GOMS
- Compare Predictive Models to Data
- Make a new *Calculator World* and PDL

GOMS

Goals, Operators, Methods, and Selection Rules
Card, Moran & Newell, 1983

- CMN (or Classic) GOMS
- NGOMSL
- KLM
- CPM

the MHP can provide time estimates for low-level operators

GOMS

Goals, Operators, Methods, and Selection Rules
Card, Moran & Newell, 1983

- CMN (or Classic) GOMS
- NGOMSL
- KLM
- CPM

we will create PDL models
of these for the ATM task
and discuss each methodology
as we go along

CMN-GOMS

- basic hierarchical goal decomposition
- only operators at the lowest level of the hierarchy take time

Blackboard Exercise: complete CMN-GOMS model for ATM

1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



The ATM Task



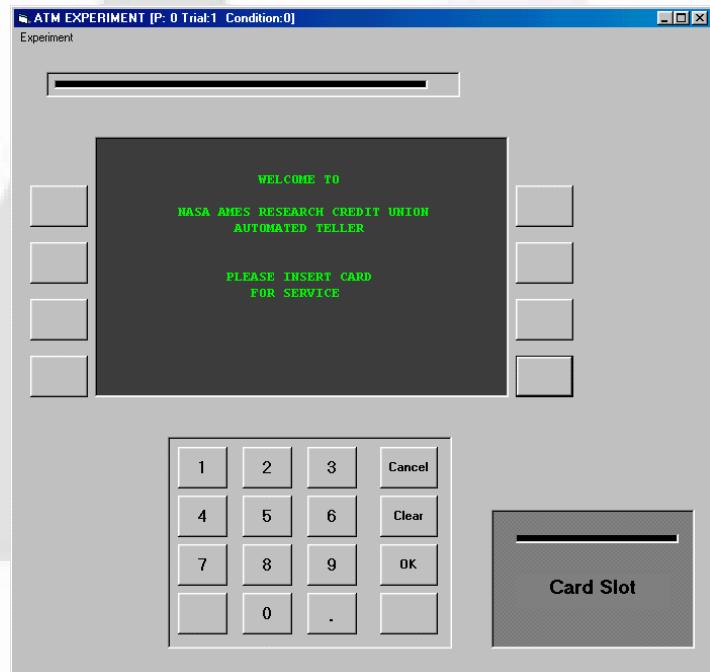
Steps:

- Insert card (click card slot)
- Enter PIN (4901)
- Press OK
- Select transaction type (withdraw)
- Select account (checking)
- Enter amount (80)
- Press if correct/not correct? (correct)
- Take cash (click cash slot)
- Other Transaction (no)
- Take card (click card slot)
- Take receipt (click cash slot)



A CMN-GOMS Model of ATM

- do banking



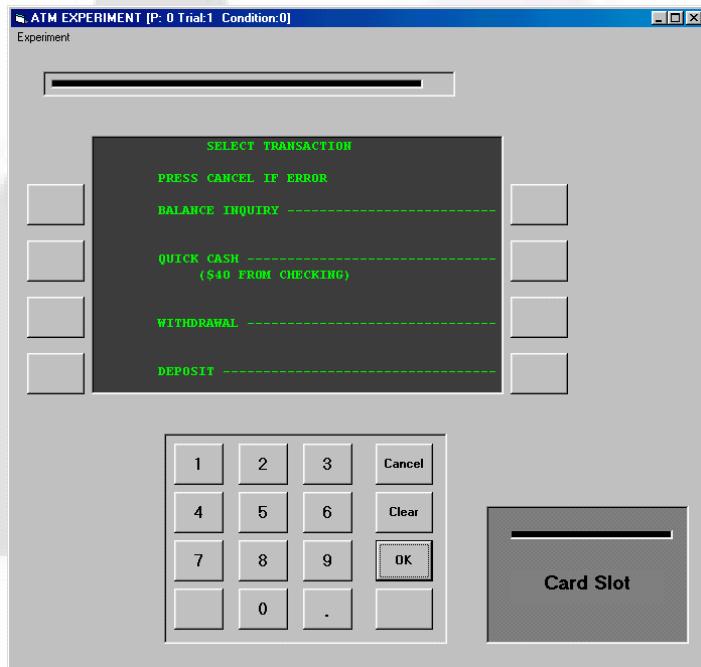
1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



A CMN-GOMS Model of ATM

- **do banking**
 - initiate session
 - do transaction
 - end session

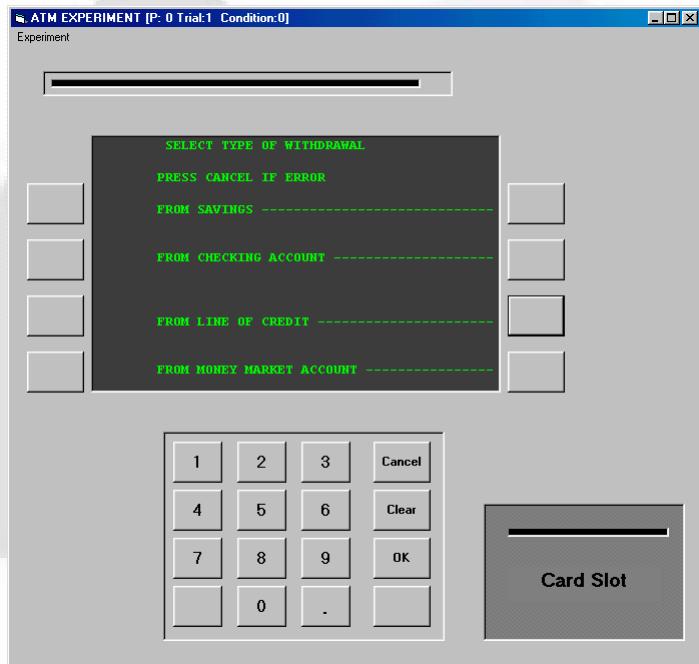


1 August 2001

Cognitive Science 2001
Edinburgh, Scotland

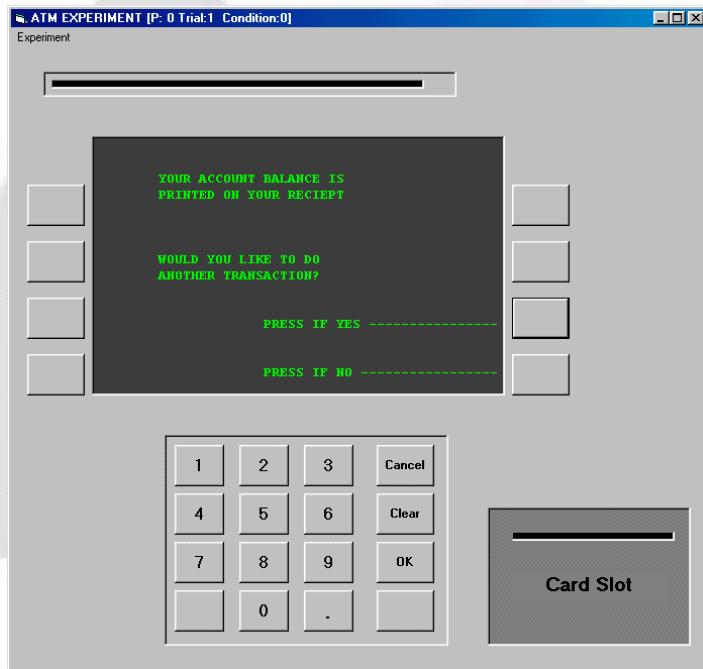


A CMN-GOMS Model of ATM



- **do banking**
 - **initiate session**
 - insert card
 - enter password
 - **do transaction**
 - choose withdraw
 - choose account
 - enter amount
 - retrieve money
 - **end session**
 - enter no [more transactions]
 - retrieve card
 - retrieve receipt

A CMN-GOMS Model of ATM



- **do banking**
 - **initiate session**
 - insert card
 - enter password
 - enter 4
 - enter 9
 - enter 0
 - enter 1
 - enter OK
 - **do transaction**
 - choose withdraw
 - choose account
 - enter amount
 - enter 8
 - enter 0
 - enter correct
 - retrieve money
 - **end session**
 - enter no [more transactions]
 - retrieve card
 - retrieve receipt

1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



PDL for CMN-GOMS

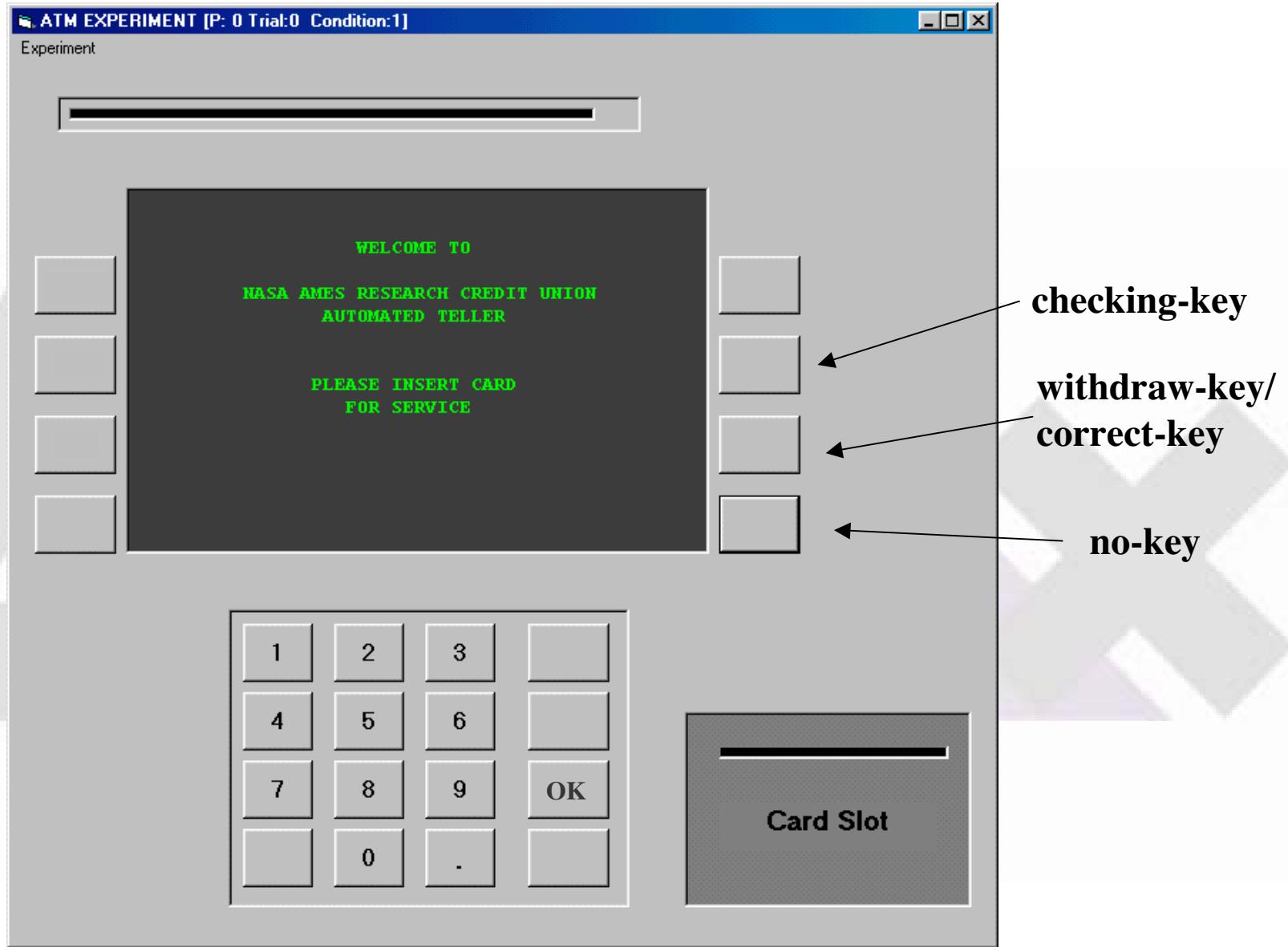
```
(procedure
  (index (initiate session))
  (step s1 (insert card))
  (step s2 (enter password) (waitfor ?s1))
  (step s4 (terminate) (waitfor ?s2)))

(procedure
  (index (insert card))
  (profile right-hand)
  (step s1 (start-activity right-hand dummy-act
    :duration 1021 => ?a))
  (step s2 (terminate) (waitfor (completed ?a))))
```

CMN ATM Model	Time*(ms)
do banking	
initiate session	
insert card	1021
enter password	
enter 4	626
enter 9	365
enter 0	420
enter 1	469
enter OK	548
do transaction	
choose withdraw	667
choose account	372
enter amount	
enter 8	684
enter 0	339
enter correct	960
retrieve money	908
end session	
enter no	935
retrieve card	856
retrieve receipt	1048
Total	10218

*times from data

Button Layout



1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



Programming Exercise: create PDL model from the CMN- GOMS model of ATM

load simworld ATM-CMN-YOURS

ATM CMN-GOMS PDL

```
(procedure
  (index (end session))
  (profile right-hand)
  (step s1 (start-activity right-hand dummy-act
    :duration 2839 => ?a))
  (step s2 (terminate) (waitfor (completed ?a))))
```

expand
leaf nodes

```
(procedure
  (index (end session))
  (step s1 ?)
  (step s2 ?)
  (step s3 ?)
  (step s4 ?))

(procedure
  (index ?)
  (profile right-hand)
  (step s1 (start-activity right-hand dummy-act
    :duration ? => ?a))
  (step s2 (terminate) (waitfor (completed ?a))))
```



ATM CMN-GOMS PDL

```
(procedure
  (index (end session))
  (profile right-hand)
  (step s1 (start-activity right-hand dummy-act
    :duration 2839 => ?a))
  (step s2 (terminate) (waitfor (completed ?a))))
```

expand
leaf nodes

```
(procedure
  (index (end session))
  (step s1 (enter no))
  (step s2 (retrieve card) (waitfor ?s1))
  (step s3 (retrieve receipt) (waitfor ?s2))
  (step s4 (terminate) (waitfor ?s3)))

(procedure
  (index (enter no))
  (profile right-hand)
  (step s1 (start-activity right-hand dummy-act
    :duration 935 => ?a))
  (step s2 (terminate) (waitfor (completed ?a))))
```



CMN

Trace

1 August 2001

Cognitive Science 2001
Edinburgh, Scotland

```
[0] (STARTED #{TASK-2 (BUILT-IN)})  
[0] (STARTED #{TASK-3 (DO-DOMAIN)})  
[0] (STARTED #{TASK-6 (DO BANKING)})  
[0] (STARTED #{TASK-4 (MAINTAIN-MEMORY-BIAS)})  
[0] (STARTED #{TASK-10 (INITIATE SESSION)})  
[0] (STARTED #{TASK-13 (INSERT CARD)})  
[0] (STARTED #{DUMMY-ACT-1 DURATION 1021})  
[1021] (STARTED #{TASK-12 (ENTER PASSWORD)})  
[1021] (STARTED #{TASK-21 (ENTER 4)})  
[1021] (STARTED #{DUMMY-ACT-2 DURATION 626})  
[1647] (STARTED #{TASK-20 (ENTER 9)})  
[1647] (STARTED #{DUMMY-ACT-3 DURATION 365})  
[2012] (STARTED #{TASK-19 (ENTER 0)})  
[2012] (STARTED #{DUMMY-ACT-4 DURATION 420})  
[2432] (STARTED #{TASK-18 (ENTER 1)})  
[2432] (STARTED #{DUMMY-ACT-5 DURATION 469})  
[2901] (STARTED #{TASK-17 (ENTER OK)})  
[2901] (STARTED #{DUMMY-ACT-6 DURATION 548})  
[3449] (STARTED #{TASK-9 (DO TRANSACTION)})  
[3449] (STARTED #{TASK-36 (CHOOSE WITHDRAW)})  
[3449] (STARTED #{DUMMY-ACT-7 DURATION 667})  
[4116] (STARTED #{TASK-35 (CHOOSE ACCOUNT)})  
[4116] (STARTED #{DUMMY-ACT-8 DURATION 372})  
[4488] (STARTED #{TASK-34 (ENTER AMOUNT)})  
[4488] (STARTED #{TASK-44 (ENTER 8)})  
[4488] (STARTED #{DUMMY-ACT-9 DURATION 684})  
[5172] (STARTED #{TASK-43 (ENTER 0*)})  
[5172] (STARTED #{DUMMY-ACT-10 DURATION 339})  
[5511] (STARTED #{TASK-42 (ENTER CORRECT)})  
[5511] (STARTED #{DUMMY-ACT-11 DURATION 960})  
[6471] (STARTED #{TASK-33 (RETRIEVE MONEY)})  
[6471] (STARTED #{DUMMY-ACT-12 DURATION 908})  
[7379] (STARTED #{TASK-8 (END SESSION)})  
[7379] (STARTED #{TASK-56 (ENTER NO)})  
[7379] (STARTED #{DUMMY-ACT-13 DURATION 935})  
[8314] (STARTED #{TASK-55 (RETRIEVE CARD)})  
[8314] (STARTED #{DUMMY-ACT-14 DURATION 856})  
[9170] (STARTED #{TASK-54 (RETRIEVE RECEIPT)})  
[9170] (STARTED #{DUMMY-ACT-15 DURATION 1048})  
[10218] (END OF RUN)
```



Programming Exercise: CMN-GOMS to KLM

load simworld ATM-KLM-YOURS

1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



CMN-GOMS to KLM

- CMN-GOMS model was built with times from our own data
- The real goal of modeling is prediction
 - It is desirable to reuse times from previous work
 - Times for the lower levels of the hierarchy generalize across different domains

KLM Modeling

- an exploration by CMN into what predictive power you could get by using only one mental operator of around 1 1/3 seconds, Fitts's Law, and three other simple motor operators with fixed times
- It was intended as a rough approximation of human performance early in practice

KLM Operators for ATM

- M Mentally prepare 1350 ms
- K Mouse Click 200 ms
- P Move Cursor Fitts's Law

A Keystroke-Level Model of the ATM Task

KLM Operators	Operator	Duration (ms)	Click Times
Initialize	M	1350	
Move pointer to slot	P	580	
Click on slot	K	200	2130
Move mouse to 4	P	330	
click mouse button	K	200	530
move mouse to 9	P	190	
click mouse button	K	200	390
move mouse to 0	P	100	
click mouse button	K	200	300
move mouse to 1	P	180	
click mouse button	K	200	380
move mouse to OK	P	230	
click mouse button	K	200	430
take out money	M	1350	
move mouse to withdraw	P	220	
click mouse button	K	200	1770
move to checking	P	60	
click mouse button	K	200	260
move to 8	P	300	
click mouse button	K	200	500
move to 0	P	50	
click mouse button	K	200	250
move to correct	P	300	
click mouse button	K	200	500
move to money	P	490	
click mouse button	K	200	690
terminate	M	1350	
move to no	P	500	
click mouse button	K	200	2050
move to money	P	370	
click mouse button	K	200	570
move to card	P	540	
click mouse button	K	200	740

Changes from CMN-GOMS

- using the right resources
 - memory for mentally prepare
 - right-hand for mouse move and mouse click
- calling out to a Lisp function to get a duration
 - Fitts's Law

KLM Operators in PDL

```
(procedure
  (index (mentally-prepare ?activity))
  (profile memory)
  (step s1 (start-activity memory memory-act :duration 1350 => ?a))
  (step s2 (terminate) (waitfor (completed ?a)))))

(procedure
  (index (mouse-click ?target))
  (profile right-hand)
  (step s1 (mouse-object => ?object))
  (step s2 (start-activity right-hand mouse-click-act :object ?object
                            :duration 200 => ?a) (waitfor ?s1))
  (step s3 (terminate) (waitfor (completed ?a))))
```



Fitts's Law in PDL

```
(procedure
  (index (move-cursor ?target))
  (profile right-hand)
  (step s1 (mouse-time ?target => ?time))
  (step s2 (start-activity right-hand mouse-move-act :object ?target
                            :duration ?time => ?a) (waitfor ?s1))
  (step s3 (terminate) (waitfor (completed ?a)))))

(procedure :special
  (index (mouse-time ?object))
  (fitts-time (pointer *mouse*) ?object))

  (setf *fitts-a* 100)
  (setf *fitts-b* 0.5)

  (defun fitts-time (pointer obj)
    (let* ((obj (lookup-unique-name obj *agent*))
           (d (distance (pos pointer) (pos obj)))
           (s (min (first (dimensions obj))
                    (second (dimensions obj)))))
      (if obj
          (floor (* *fitts-a* (log (+ (/ d s) *fitts-b*) 2)))
          (format t "Error: no object found~&")))))
```



KLM PDL

```
(procedure  
  (index (end session))  
  (step t (terminate)))
```

use KLM
operators

```
(procedure  
  (index (end session))  
  (step m1 ?)  
  (step p1 ?)  
  (step k1 ?)  
  (step p2 ?)  
  (step k2 ?)  
  (step p3 ?)  
  (step k3 ?)  
  (step t (terminate) (waitfor ?k3)))
```



KLM PDL

```
(procedure  
  (index (end session))  
  (step t (terminate)))
```

use KLM
operators

```
(procedure  
  (index (end session))  
  (step m1 (mentally-prepare do-transaction))  
  (step p1 (move-cursor no-key) (waitfor ?m1))  
  (step k1 (mouse-click no-key) (waitfor ?p1))  
  (step p2 (move-cursor card-slot) (waitfor ?k1))  
  (step k2 (mouse-click card-slot) (waitfor ?p2))  
  (step p3 (move-cursor money-slot) (waitfor ?k2))  
  (step k3 (mouse-click money-slot) (waitfor ?p3))  
  (step t (terminate) (waitfor ?k3)))
```

KLM and CPM-GOMS

- Baskin & John (1998) found that KLM, a serial resource model, is a useful approximation of early skill
- They also found that CPM-GOMS, a model that assumes resource parallelism at the lowest level, predicted behavior well at around 100 trials of practicing the skill

MHP

- Model Human Processor (CMN, 1983)
 - separate processors for perception, cognition, motor
 - each with associated cycle times

ATM Study



- 2 subjects
- 200 trials each

Steps:

- Insert card (click card slot)
- Enter PIN (4901)
- Press OK
- Select transaction type (withdraw)
- Select account (checking)
- Enter amount (80)
- Press if correct/not correct? (correct)
- Take cash (click cash slot)
- Other Transaction (no)
- Take card (click card slot)
- Take receipt (click cash slot)

1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



Models and Humans

	Subject 1		
	10th trial	99th trial	195th trial
CARDSLOT	1371	1354	981
4-KEY	1058	616	528
9-KEY	431	424	339
0-KEY	375	369	291
1-KEY	512	426	400
OK	707	535	483
WITHDRAW	981	586	539
CHECKING	396	260	260
8-KEY	795	749	664
0-KEY	373	318	261
CORRECT	833	720	582
CASHSLOT	1864	1167	598
NO	1267	637	508
CARDSLOT	882	689	660
CASHSLOT	1140	971	1025
TOTAL TIM	12,985	9,821	8,119

	Subject 2		
	9th trial	94th trial	196th trial
CARDSLOT	1130	1021	1049
4-KEY	888	626	664
9-KEY	528	365	315
0-KEY	417	420	369
1-KEY	592	469	420
OK	619	548	609
WITHDRAW	1102	667	592
CHECKING	494	372	372
8-KEY	1083	684	1416
0-KEY	396	339	288
CORRECT	775	960	701
CASHSLOT	1480	908	653
NO	981	935	659
CARDSLOT	774	856	991
CASHSLOT	1133	1048	733
TOTAL TIM	12,392	10,218	9,831

	KLM Model	CPM Model
CARDSLOT	2201	1001
4-KEY	509	709
9-KEY	392	492
0-KEY	329	520
1-KEY	412	520
OK	452	720
WITHDRAW	1832	582
CHECKING	289	470
8-KEY	534	634
0-KEY	275	375
CORRECT	532	782
CASHSLOT	806	1056
NO	1922	822
CARDSLOT	655	905
CASHSLOT	840	1090
TOTAL TIM	11,980	10,678

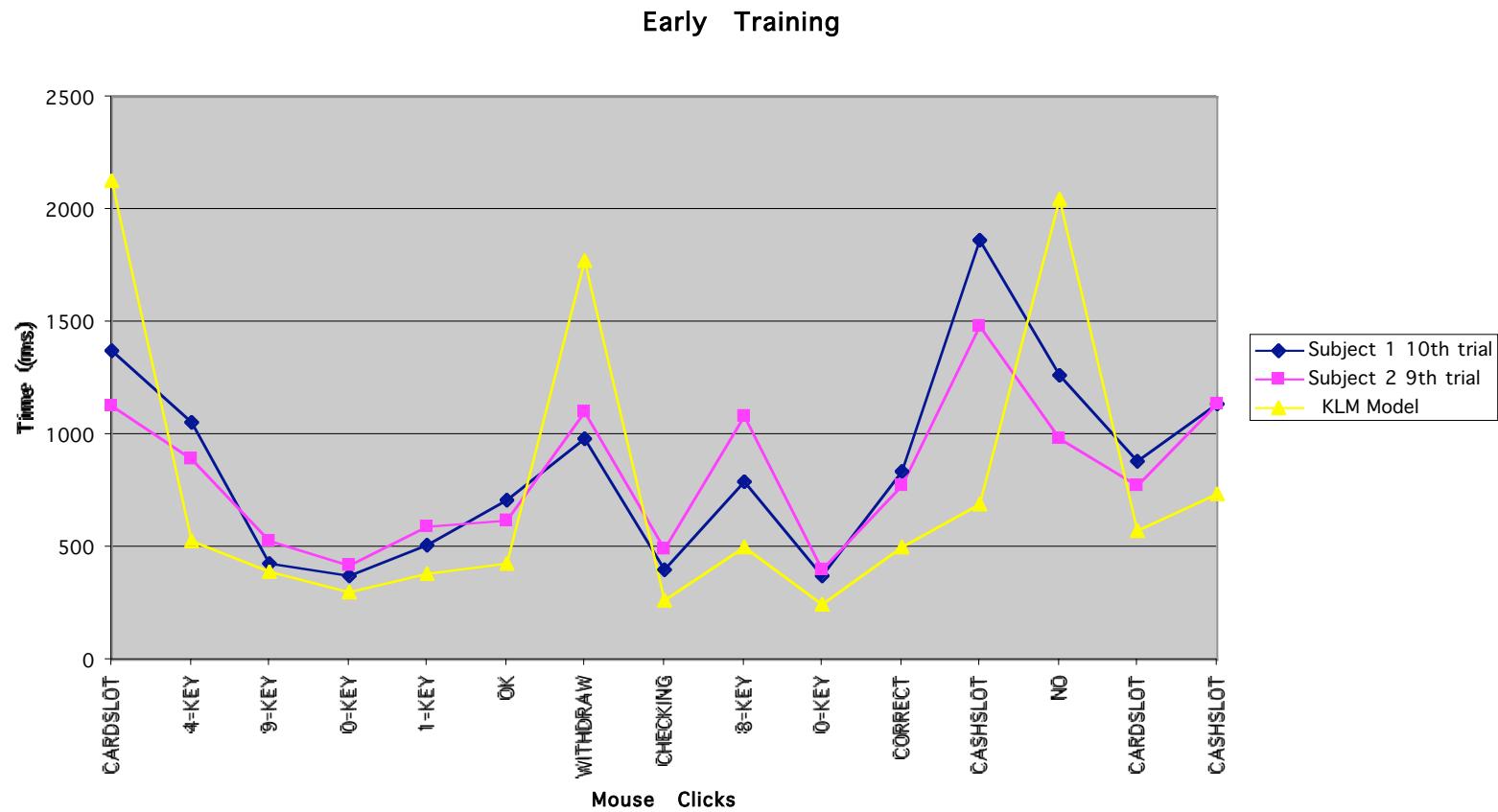
-see Baskin & John, 1998
for similar results

1 August 2001

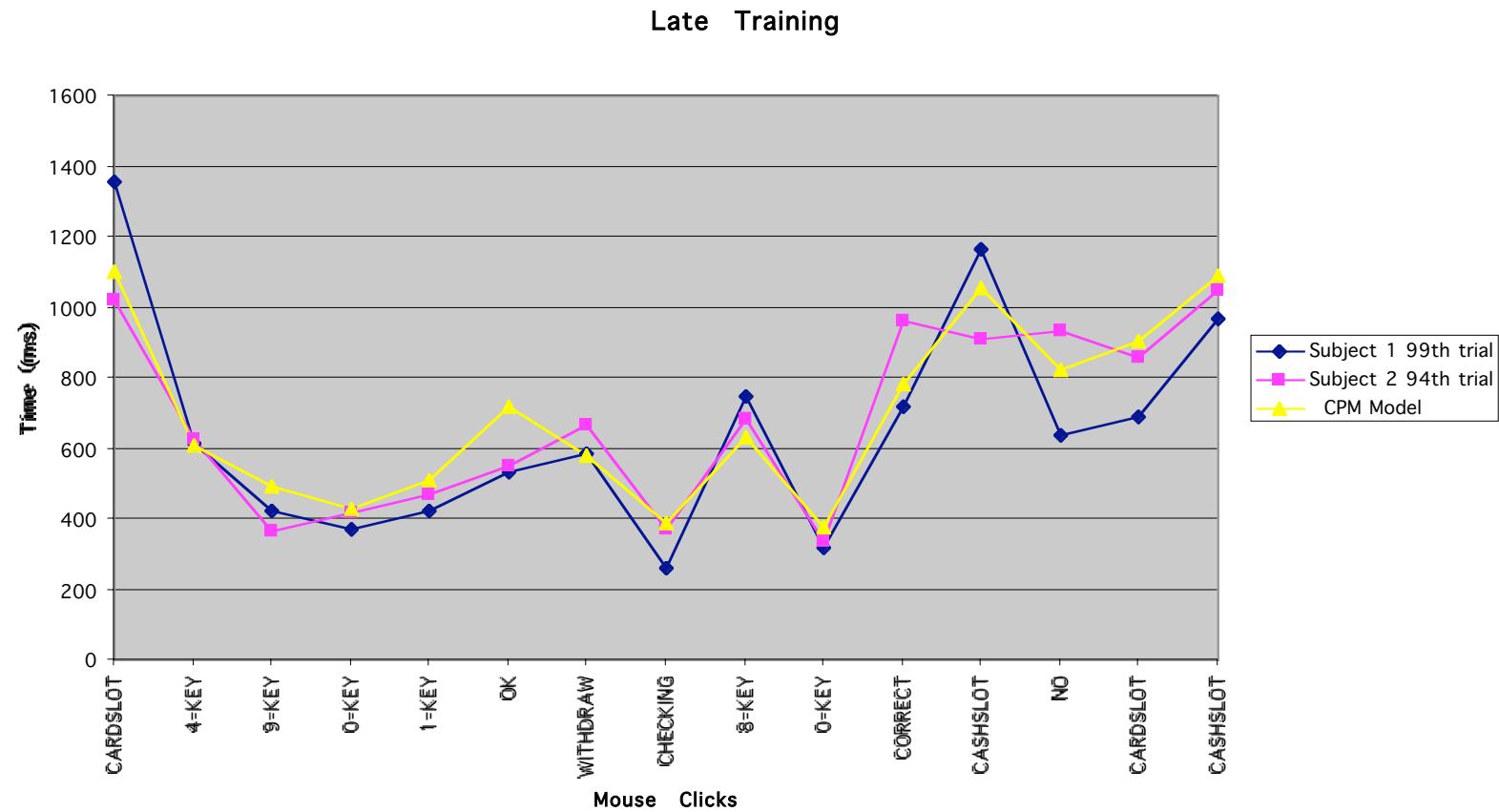
Cognitive Science 2001
Edinburgh, Scotland



Subjects Early in Practice vs. KLM



Subjects Late in Practice vs. CPM-GOMS



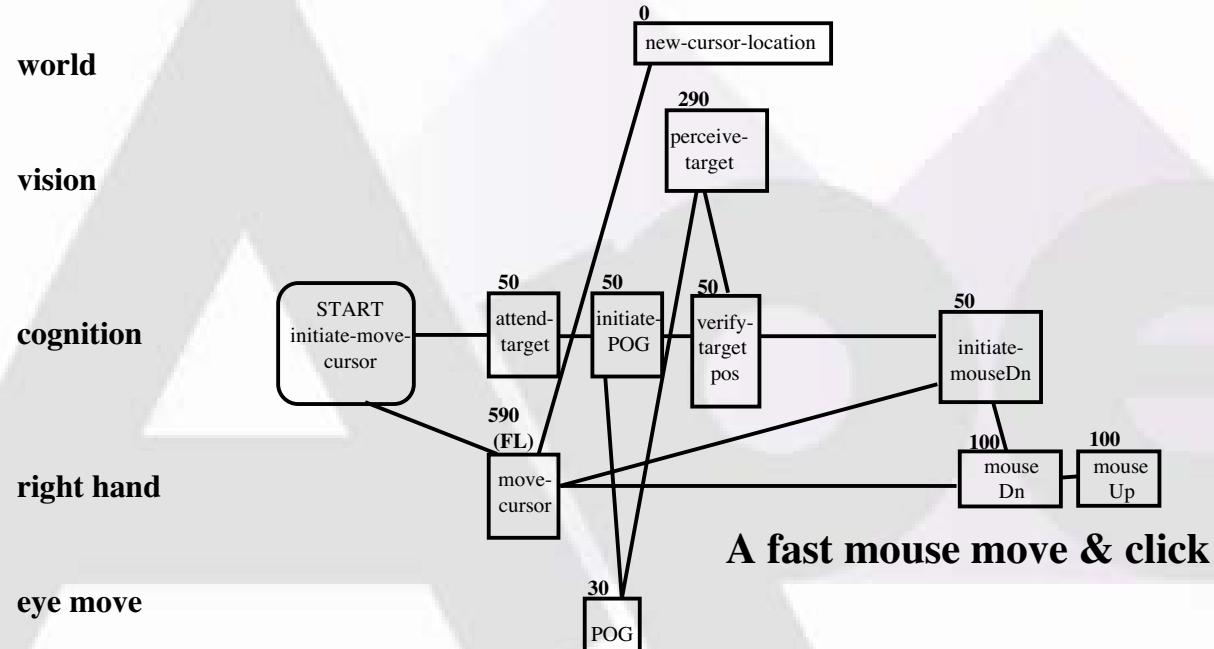
-see Baskin & John, 1999
for similar results



CPM-GOMS

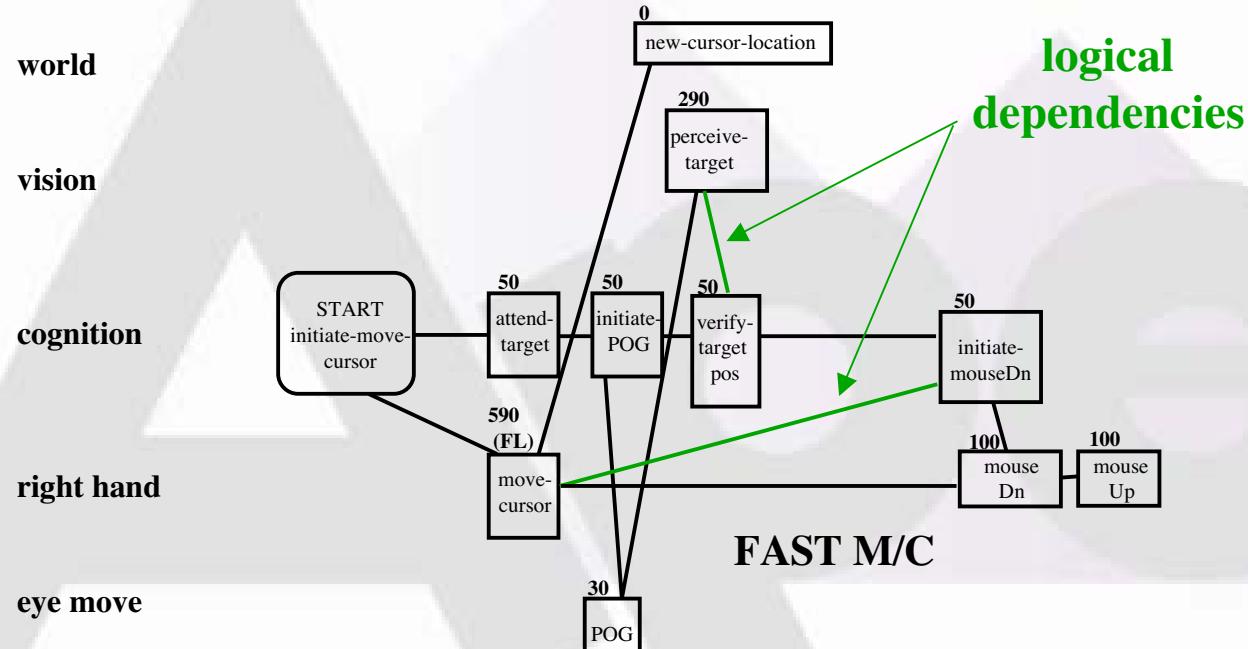
- Resource parallelism: the ability to engage in more than one mode of behavior at once (e.g. eyes and hands)
- CPM-GOMS joins CMN-GOMS with the MHP utilizing resource parallelism for steps at the lowest levels in the hierarchy
- These steps must fully specify what resources they require

Using Parallel Resources



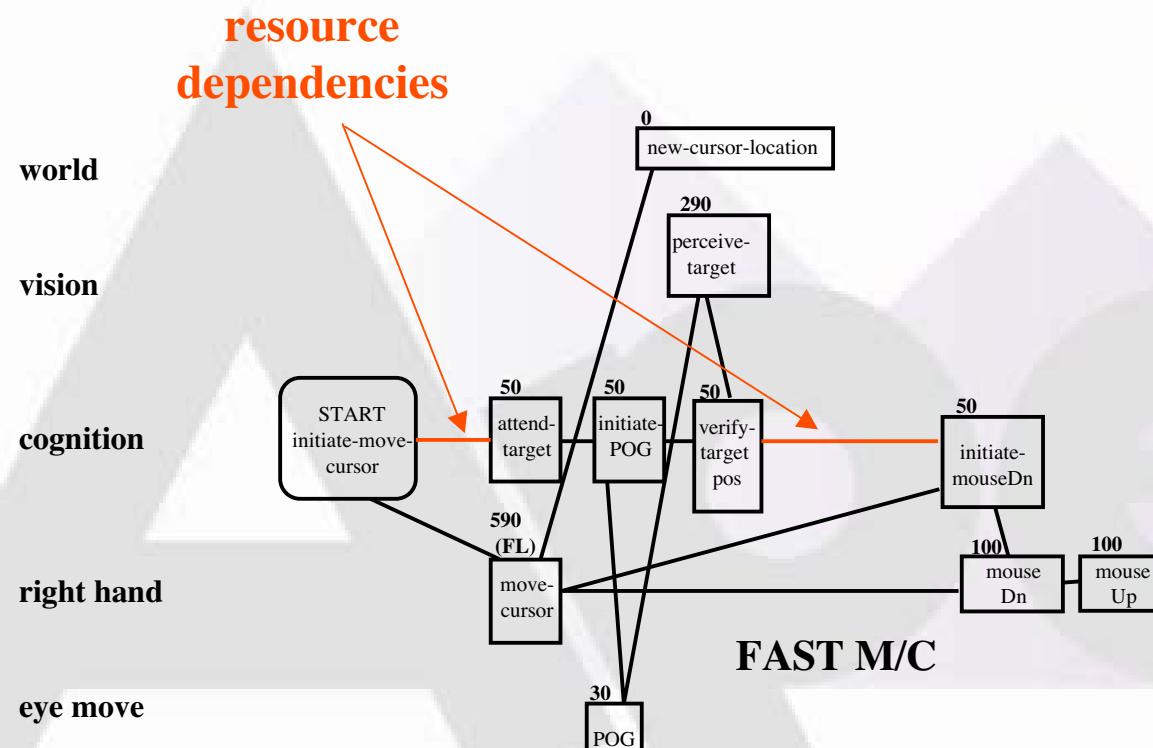
from *Gray and
Boehm-Davis (2000)*

Logical Dependencies



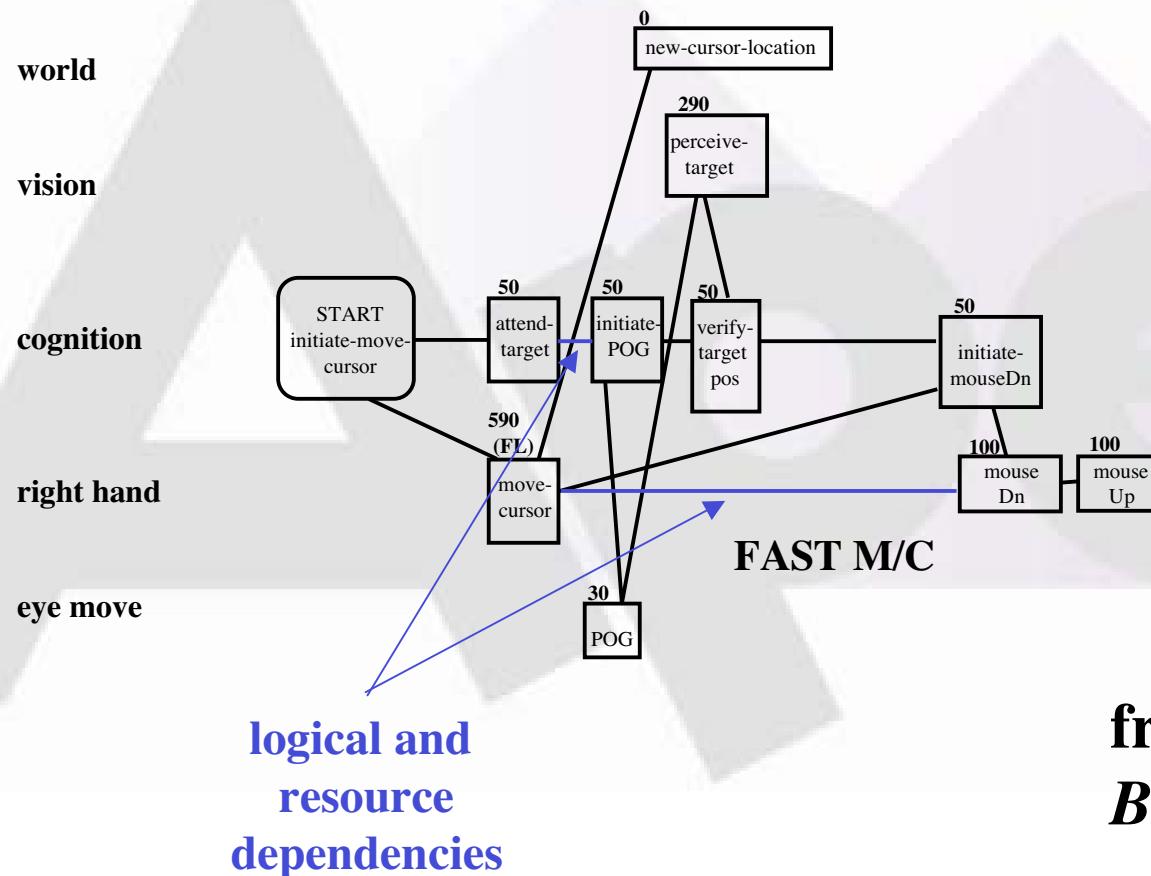
from *Gray and
Boehm-Davis (2000)*

Resource Dependencies



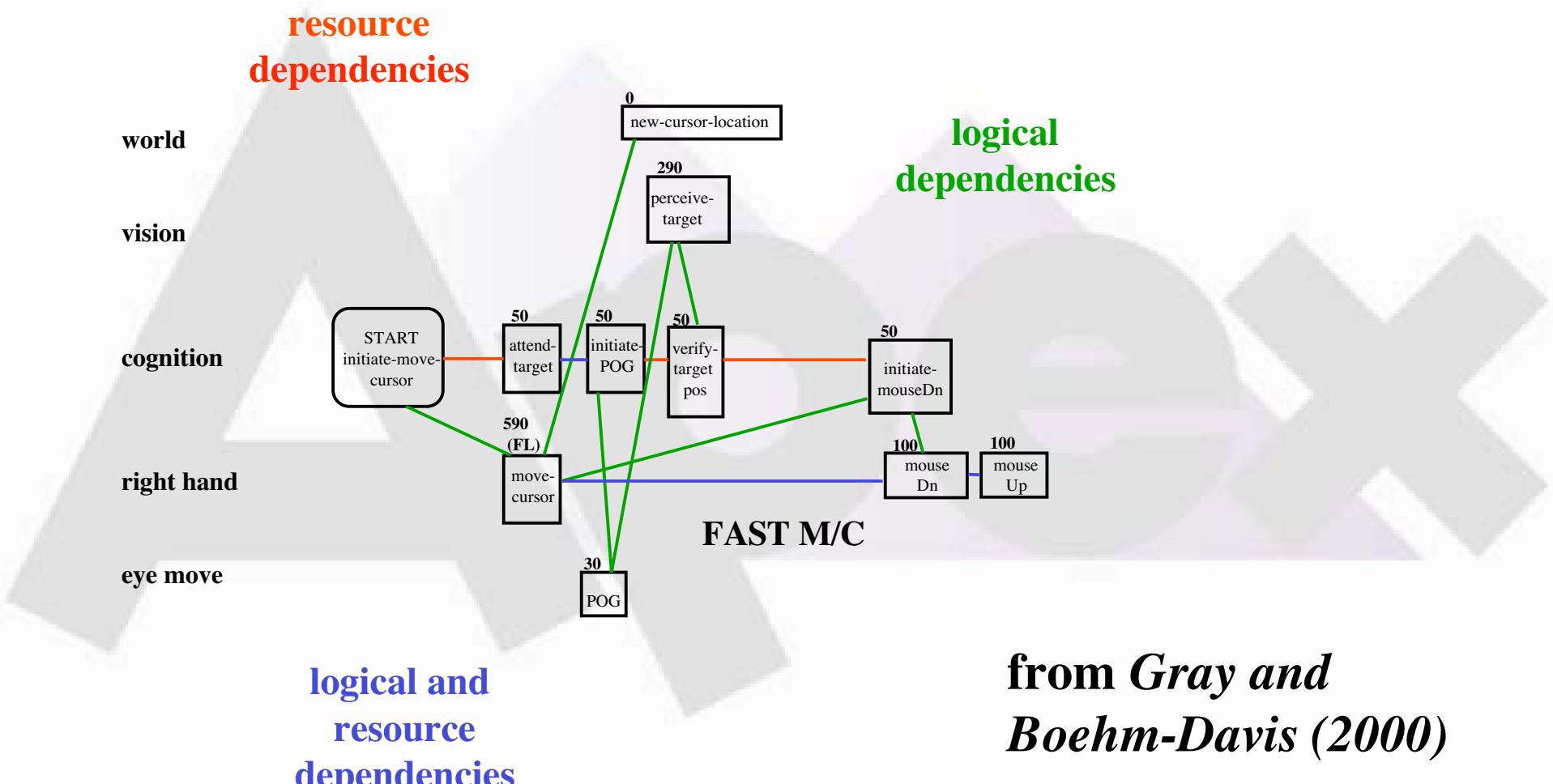
from *Gray and
Boehm-Davis (2000)*

Double Dependencies



from *Gray and
Boehm-Davis (2000)*

All the Dependencies



1 August 2001

Cognitive Science 2001
Edinburgh, Scotland

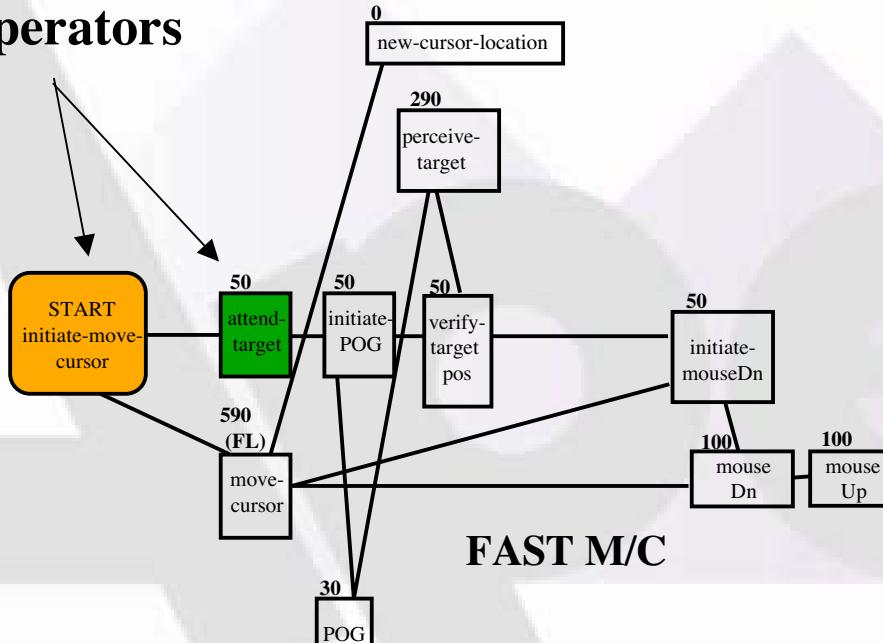


Modeling with Templates

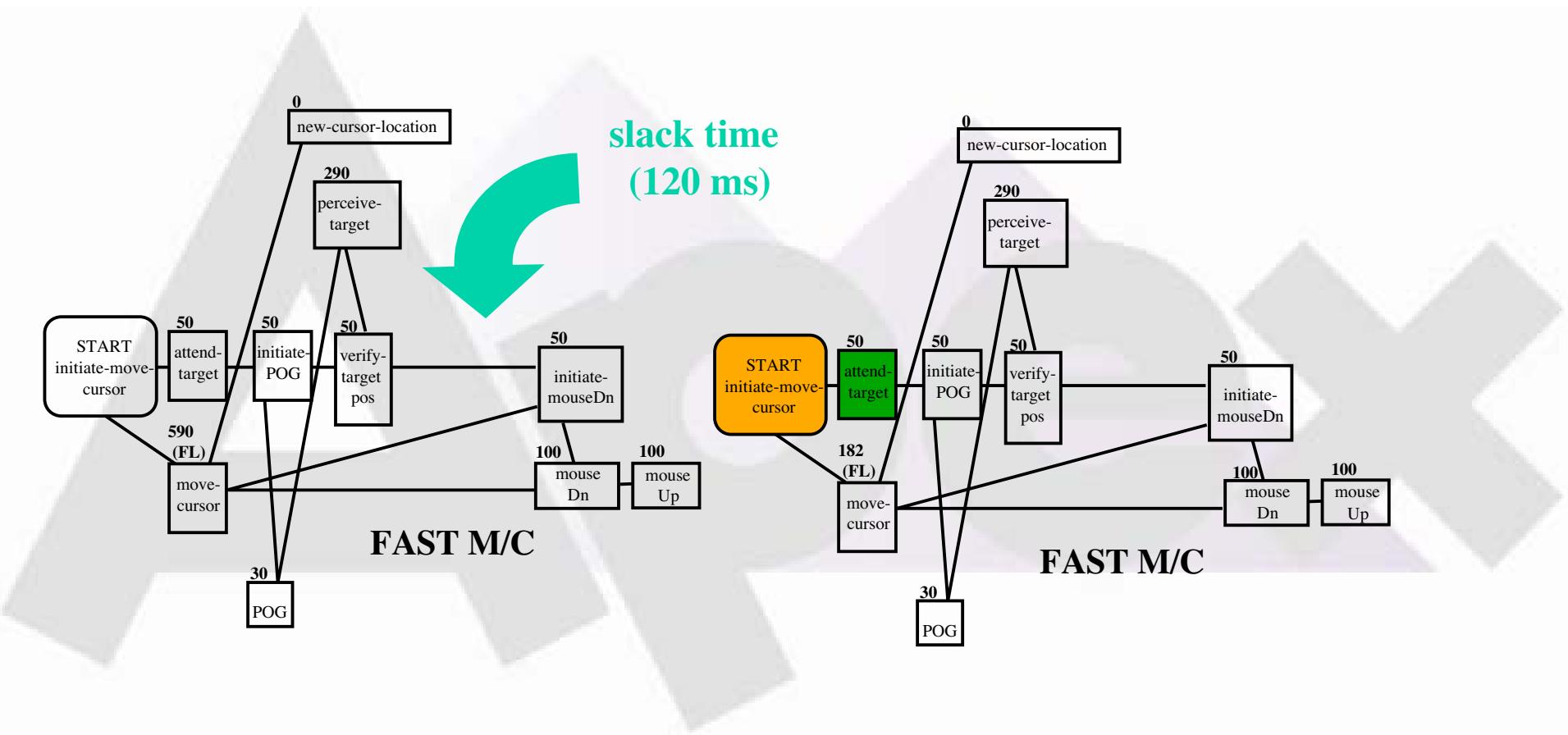
- To make a model of a task like withdrawing money from an ATM, you need to string together the appropriate templates
- and then it gets hard...

Interleaving Templates

**within a template these do
not have logical dependencies
on any other operators**



Interleaving Templates

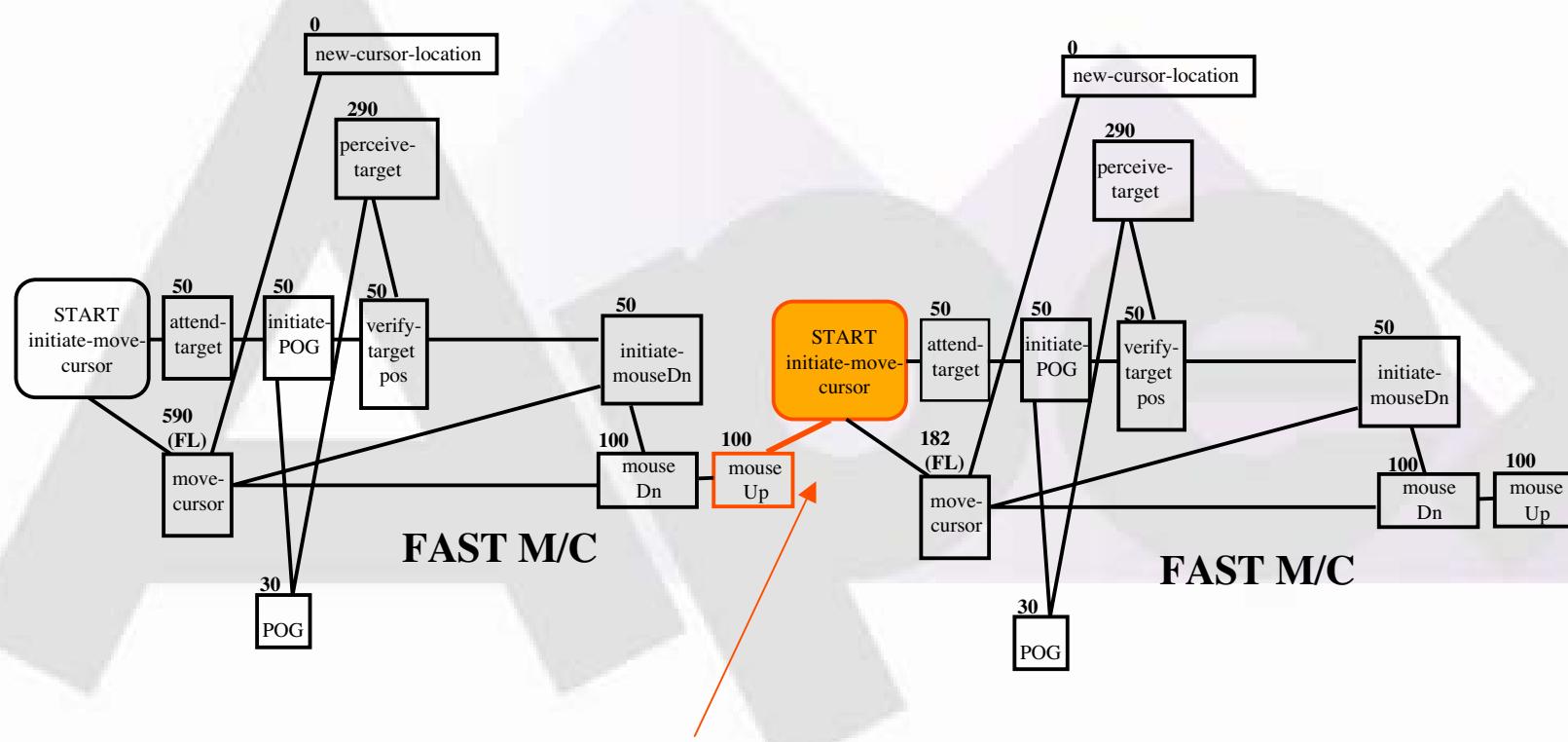


1 August 2001

Cognitive Science 2001
Edinburgh, Scotland

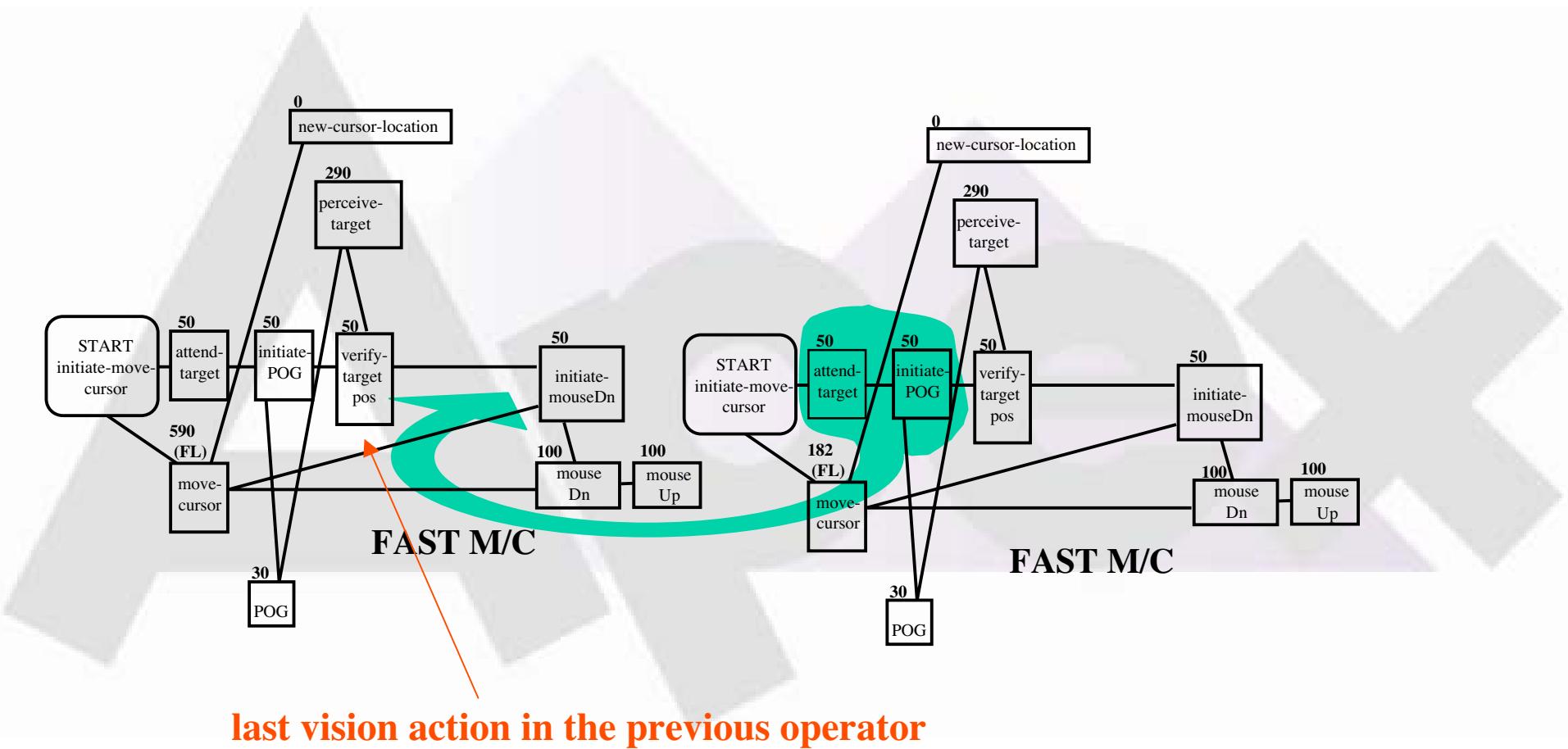


Interleaving Templates

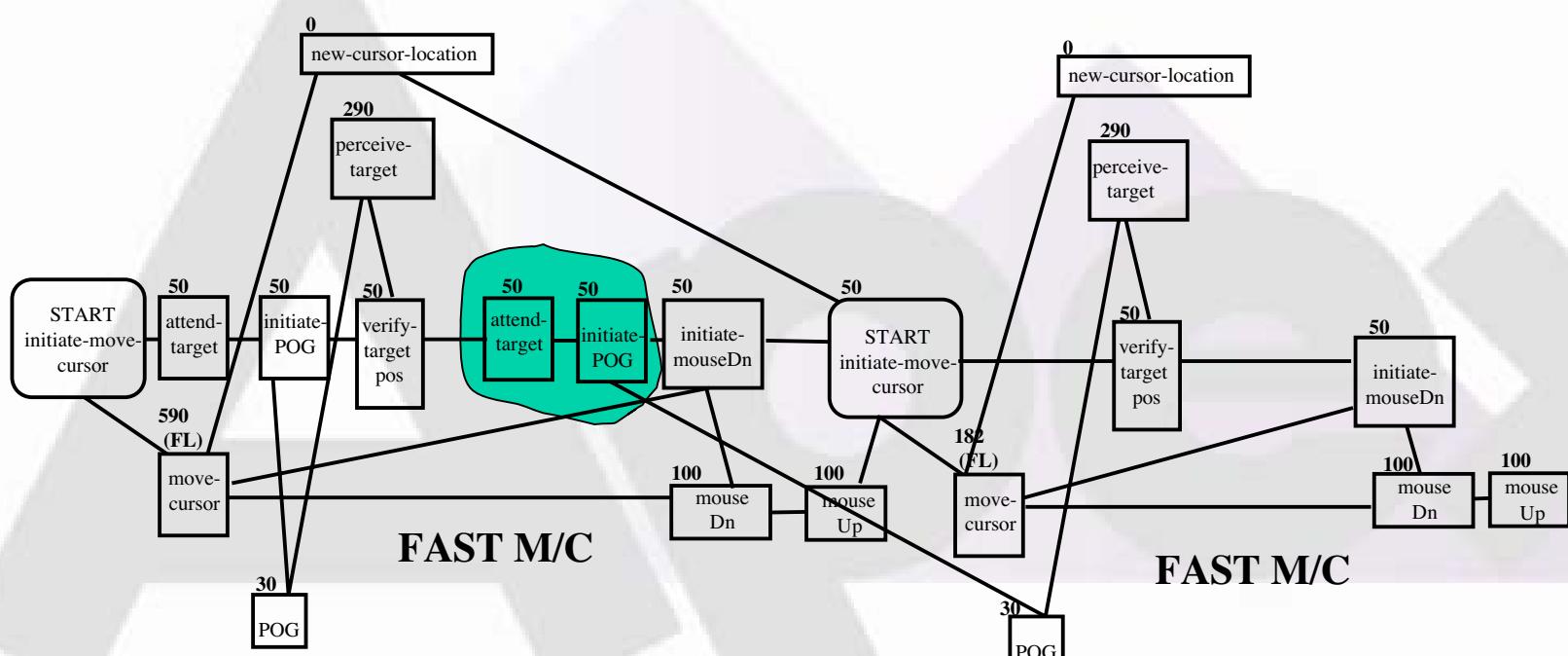


initiate move cursor cannot precede the last motor action with the same hand in the previous operator

Interleaving Templates



Interleaving Templates



Summary of Interleaving Procedure

At each boundary between templates, ask...

1. Is there enough slack time at the end of the first template to allow any of the cognitive operators at the beginning of the next template to interleave? If so,
2. Are there any logical dependencies preventing a candidate cognitive operator from interleaving?
If not,
3. Interleave the candidate operator and GOTO 1.

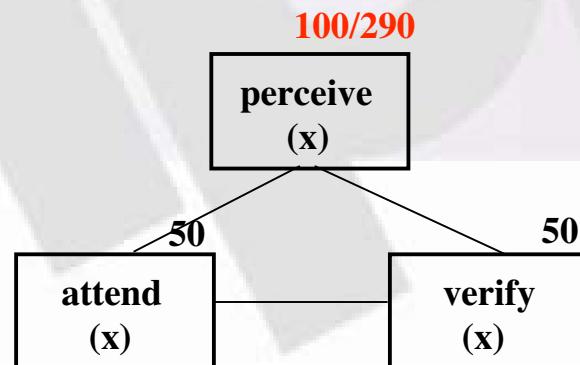
How to Make a CPM-GOMS Model in 5 Hours or More

- Create serial CMN-GOMS model with templates at bottom of hierarchy
- By hand, Copy&Paste the appropriate sequence of pre-created templates into MacProject
 - ATM task needs 15 templates, about 15 screensful, 10 pages printed out
- Think hard about interleaving at each boundary of the templates
 - By hand, insert duration parameters and sensible names, remove and redo a lot of dependencies (and try not to make mistakes).

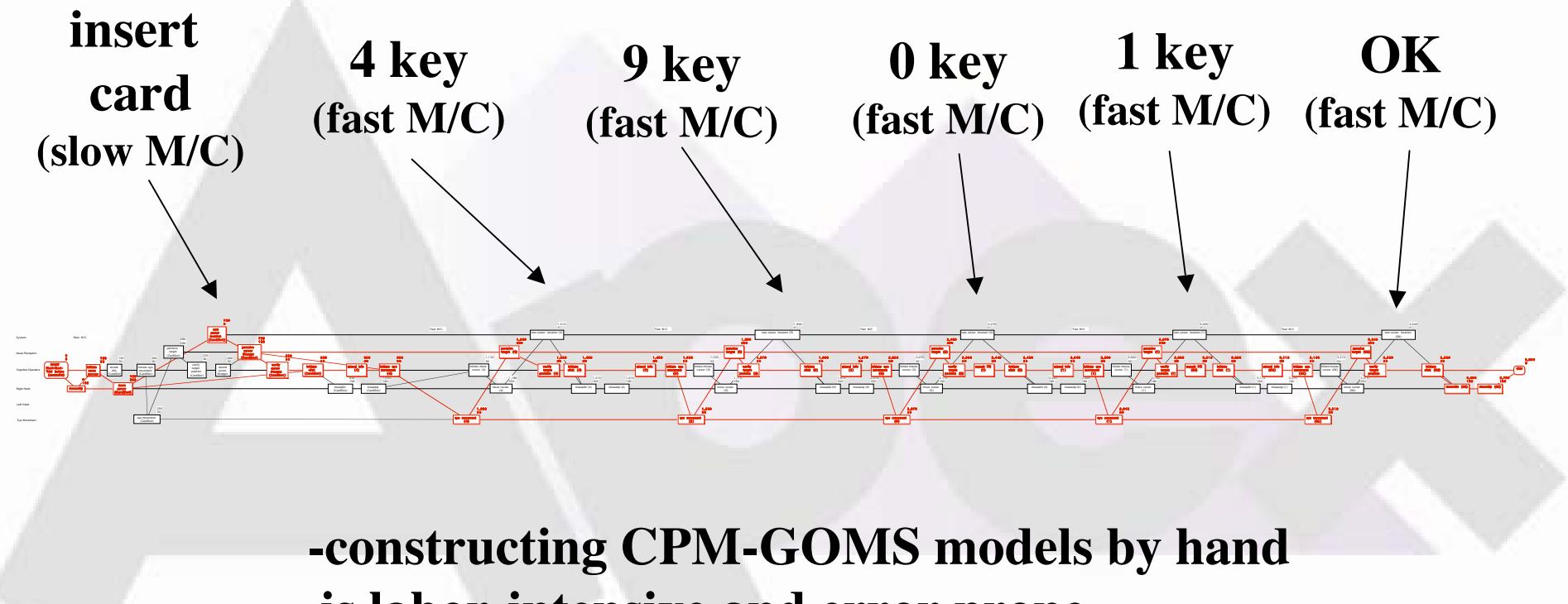


How to Make a CPM-GOMS Model in 5 Hours or More

- Two other things that make doing CPM-GOMS by hand onerous
 - changing duration parameters
 - filling in names of operator targets (x)



A CPM-GOMS Model of the ATM Task



**-constructing CPM-GOMS models by hand
is labor-intensive and error prone
(this part of the model took over 6 hours
with MacProject)**



How Apex does it in 5 Minutes or Less

- Create serial CMN-GOMS model with templates at bottom of hierarchy
- Implement the CMN in PDL, simply calling procedures that contain pre-created templates at the bottom
- Run the PDL program
- Press the PERT-chart button in Sherpa

Programming Exercise: CMN-GOMS to CPM-GOMS

load simworld ATM-CPM-YOURS

1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



Changes from KLM

- priorities
- templates
- using resources in parallel
 - vision for perception
 - memory for cognition
 - right-hand for mousing
 - gaze for eye movement
 - virtual resources: holding and releasing



Using Priorities to Turn CMN to CPM GOMS

```
(procedure
  (index (do-domain))
  (step s1 (do banking))
  (step s2 (stop) (waitfor ?s1)))

(procedure
  (index (do banking))
  (step s1 (initiate session) (priority 300))
  (step s2 (do transaction) (priority 200))
  (step s3 (end session) (priority 100))
  (step s4 (terminate) (waitfor ?s3  ?s2  ?s1)))
```

priorities allow
soft-sequencing of steps
i.e., steps are carried out
in the given order but
their sub-components
can interleave

higher priorities
go first



Goal Decomposition with Priorities

```
(procedure
  (index (initiate session))
  (step s1 (insert card) (priority 320))
  (step s2 (enter password) (priority 310))
  (step s4 (terminate) (waitfor ?s2 ?s1)))

(procedure
  (index (do transaction))
  (step s1 (choose withdraw) (priority 240))
  (step s2 (choose account) (priority 230))
  (step s3 (enter amount) (priority 220))
  (step s4 (retrieve money) (priority 210))
  (step s5 (terminate)
    (waitfor ?s4 ?s3 ?s2 ?s1)))
```

}

(initiate session)
(priority 300)

}

(do transaction)
(priority 200)



The Rest of the Priorities

```
(procedure
  (index (end session))
  (step s1 (enter-NO) (priority 130))
  (step s2 (retrieve card) (priority 120))
  (step s3 (retrieve receipt) (priority 110))
  (step s4 (terminate) (waitfor ?s3 ?s2 ?s1)))

(procedure
  (index (enter password))
  (step s1 (remember PIN) (priority 316))
  (step s2 (enter-number 4-key) (priority 315))
  (step s3 (enter-number 9-key) (priority 314))
  (step s4 (enter-number 0-key) (priority 313))
  (step s5 (enter-number 1-key) (priority 312))
  (step s6 (enter-OK) (priority 311))
  (step s7 (terminate) (waitfor ?s6 ?s5 ?s4 ?s3 ?s2 ?s1)))

(procedure
  (index (enter amount))
  (step s1 (enter-number 8-key) (priority 223))
  (step s2 (enter-number 0-key) (priority 222))
  (step s3 (enter-CORRECT) (priority 221))
  (step s4 (terminate) (waitfor ?s2 ?s1 ?s3)))
```



One More Level Down

```
(procedure
  (index (insert card))
  (step s1 (slow-move-click card-slot))
  (step s2 (terminate) (waitfor ?s1)))

(procedure
  (index (choose withdraw))
  (step s1 (fast-move-click withdraw-key))
  (step s2 (terminate) (waitfor ?s1)))

(procedure
  (index (choose account))
  (step s1 (fast-move-click checking-key))
  (step s2 (terminate) (waitfor ?s1)))

(procedure
  (index (retrieve money))
  (step s1 (slow-move-click money-slot))
  (step s2 (terminate) (waitfor ?s1)))
```

**every low-level step
calls a template**



Templates

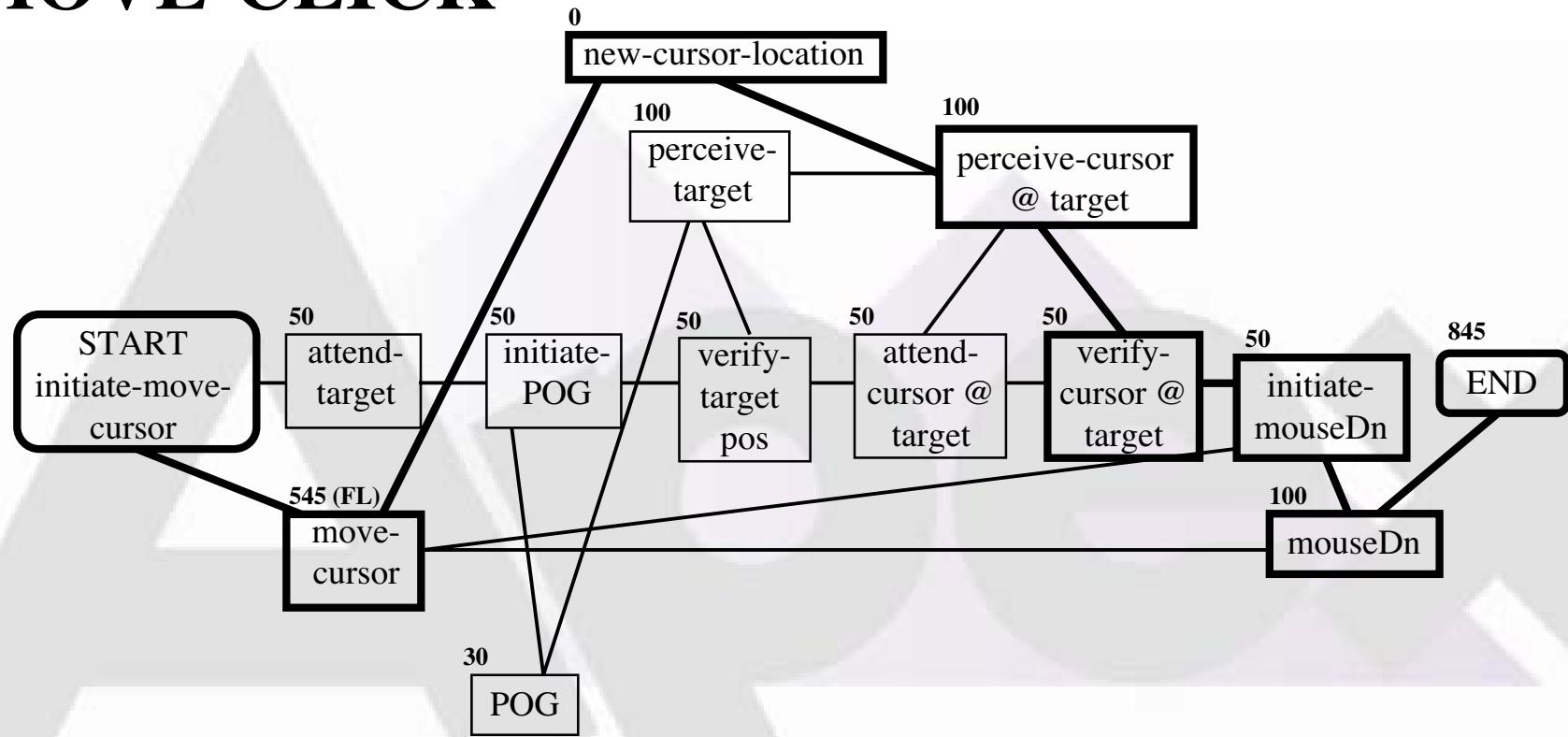
```
(procedure
  (index (slow-move-click ?target))
  (step c1 (initiate-move-cursor ?target))
  (step m1 (move-cursor ?target)
          (waitfor ?c1))
  (step c2 (attend-target ?target))
  (step c3 (initiate-eye-movement ?target)
          (waitfor ?c2))
  (step m2 (eye-movement ?target)
          (waitfor ?c3))
  (step p1 (perceive-target-complex ?target)
          (waitfor ?m2))
  (step c4 (verify-target-position ?target)
          (waitfor ?c3 ?p1))
  (step c5 (attend-cursor-at-target ?target)
          (waitfor ?c4))
  (step w1 (WORLD new-cursor-location ?target)
          (waitfor ?m1))
  (step p2 (perceive-cursor-at-target ?target)
          (waitfor ?p1 ?c5 ?w1))
  (step c6 (verify-cursor-at-target ?target)
          (waitfor ?c5 ?p2))
  (step c7 (initiate-click ?target)
          (waitfor ?c6 ?m1))
  (step m3 (mouse-down ?target)
          (waitfor ?m1 ?c7))
  (step m4 (mouse-up ?target)
          (waitfor ?m3)))
  (step t1 (terminate)
          (waitfor ?m4))))
```

templates are slightly modified versions of
Slow M/C and **Fast M/C**
from Gray & Boehm-Davis (2000)



SLOW MOVE-CLICK

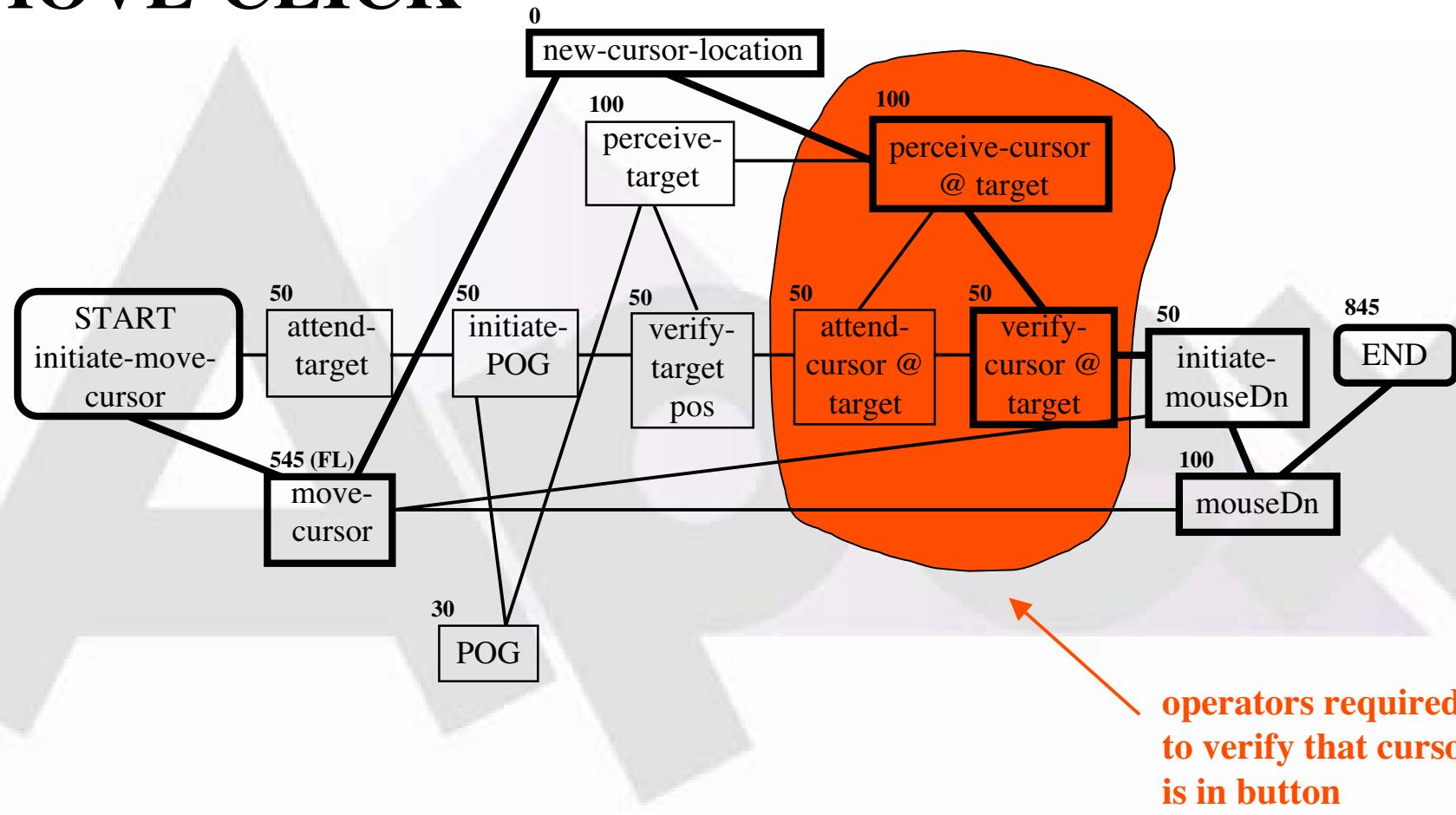
REMINDER



- from *Gray and Boehm-Davis (2000)*

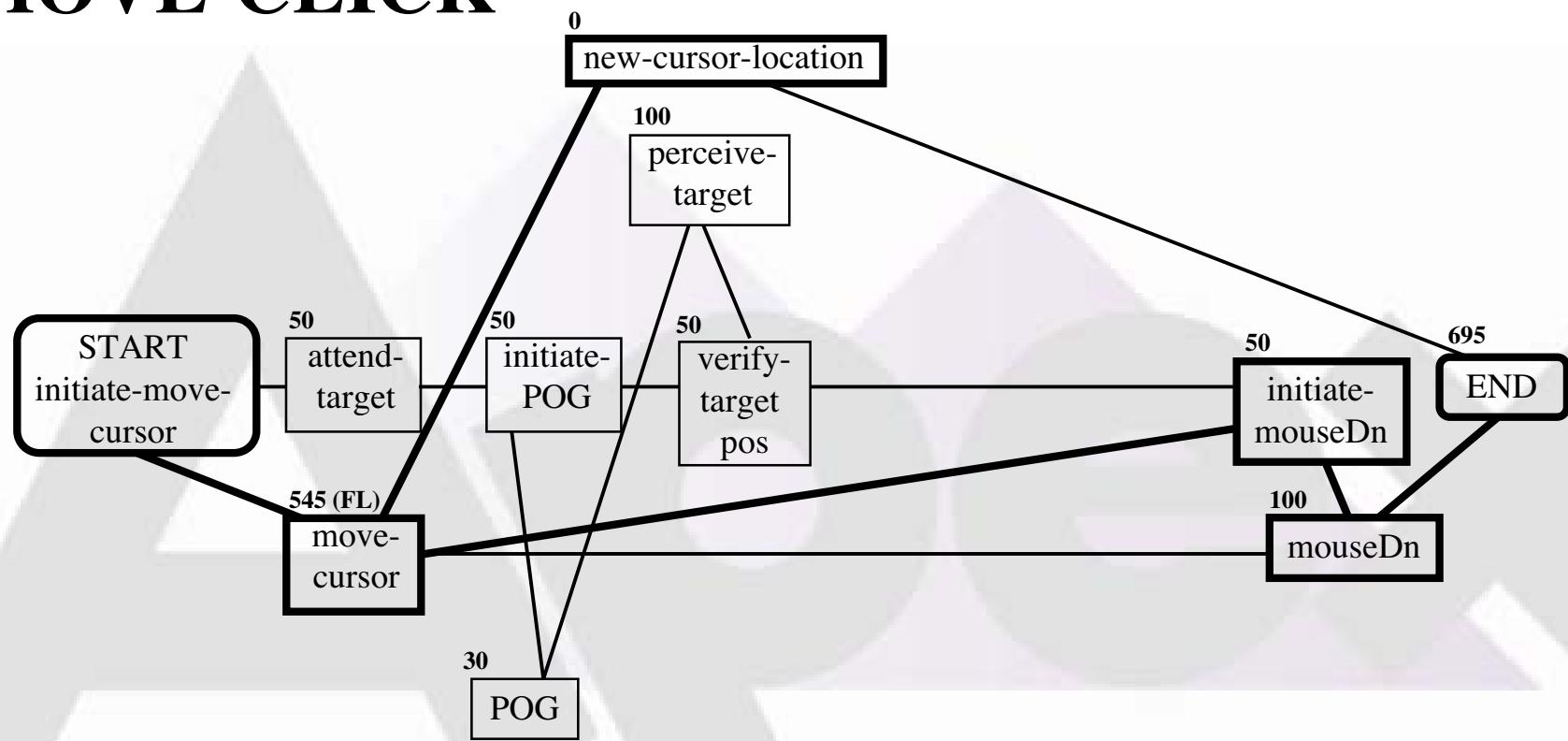
SLOW MOVE-CLICK

REMINDER



FAST MOVE-CLICK

REMINDER



- from *Gray and Boehm-Davis (2000)*

Templates

```
(procedure
  (index (fast-move-click ?target))
  (step c1 (initiate-move-cursor ?target))
  (step m1 (move-cursor ?target)
         (waitfor ?c1))
  (step c2 (attend-target ?target))
  (step c3 (initiate-eye-movement ?target)
         (waitfor ?c2))
  (step m2 (eye-movement ?target)
         (waitfor ?c3))
  (step p1 (perceive-target-complex-RR ?target)
         (waitfor ?m2))
  (step c4 (verify-target-position ?target)
         (waitfor ?c3 ?p1))
  (step w1 (WORLD new-cursor-location ?target)
         (waitfor ?m1))
; (step c5 (attend-cursor-at-target ?target)
;           (waitfor ?c4))
; (step p2 (perceive-cursor-at-target ?target)
;           (waitfor ?p1 ?c5 ?w1))
; (step c6 (verify-cursor-at-target ?target)
;           (waitfor ?c5 ?p2))
  (step c7 (initiate-click ?target)
         (waitfor ?c4 ?m1))
  (step m3 (mouse-down ?target)
         (waitfor ?m1 ?c7))
  (step m4 (mouse-up ?target)
         (waitfor ?m3))
  (step t1 (terminate)
         (waitfor ?m4))))
```

templates are slightly modified versions of **Slow M/C** and **Fast M/C** from Gray & Boehm-Davis (2000)

the **initiate-click** operator now waits for **verify-target-position**



How Apex Interleaves CPM- GOMS Templates

1 August 2001

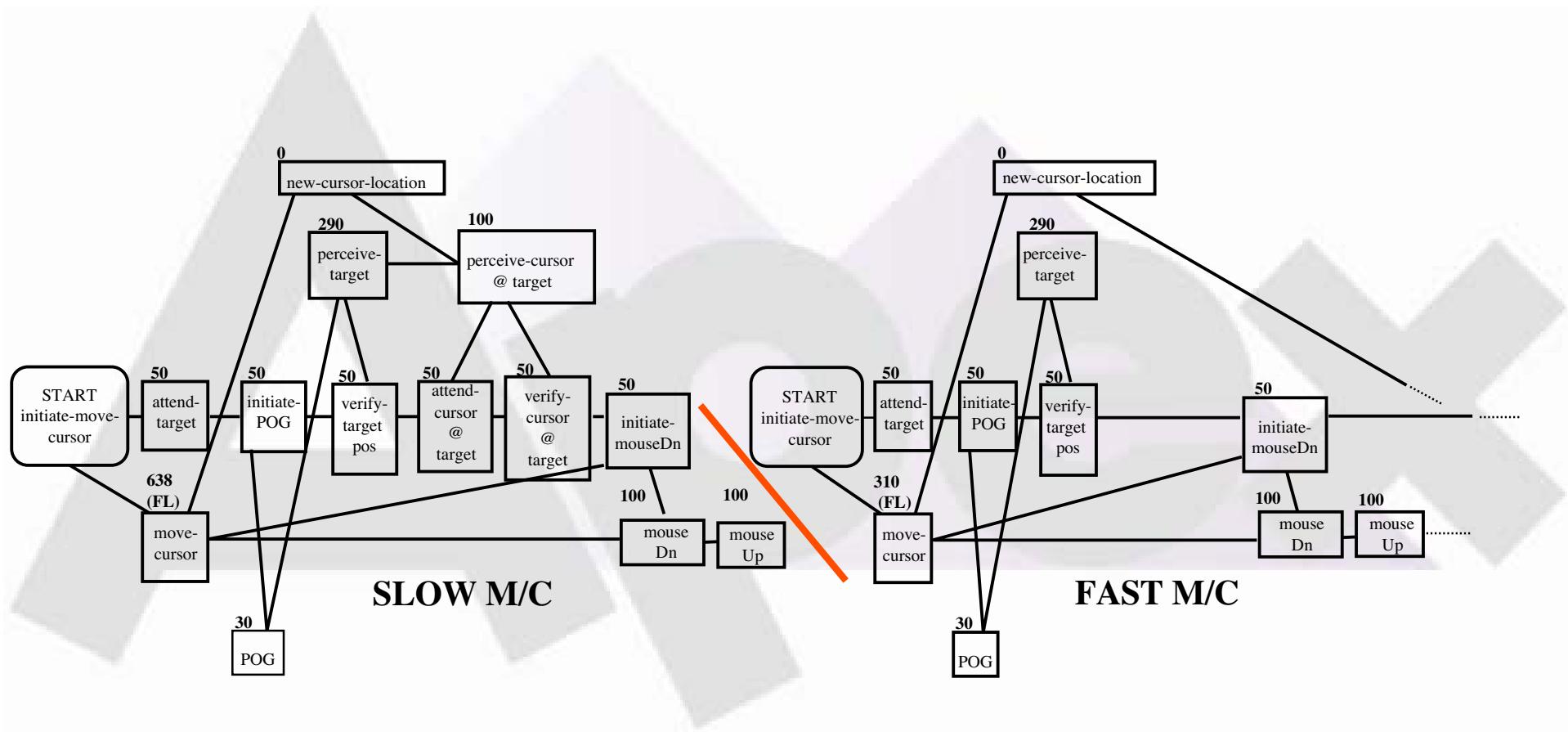
Cognitive Science 2001
Edinburgh, Scotland



Remember the hard part?

- Are there any logical dependencies preventing a candidate cognitive operator from interleaving?
- Apex's mechanisms demystify the “art” it used to be when interleaving was done by hand in MacProject

Interleaving Templates w/ Apex

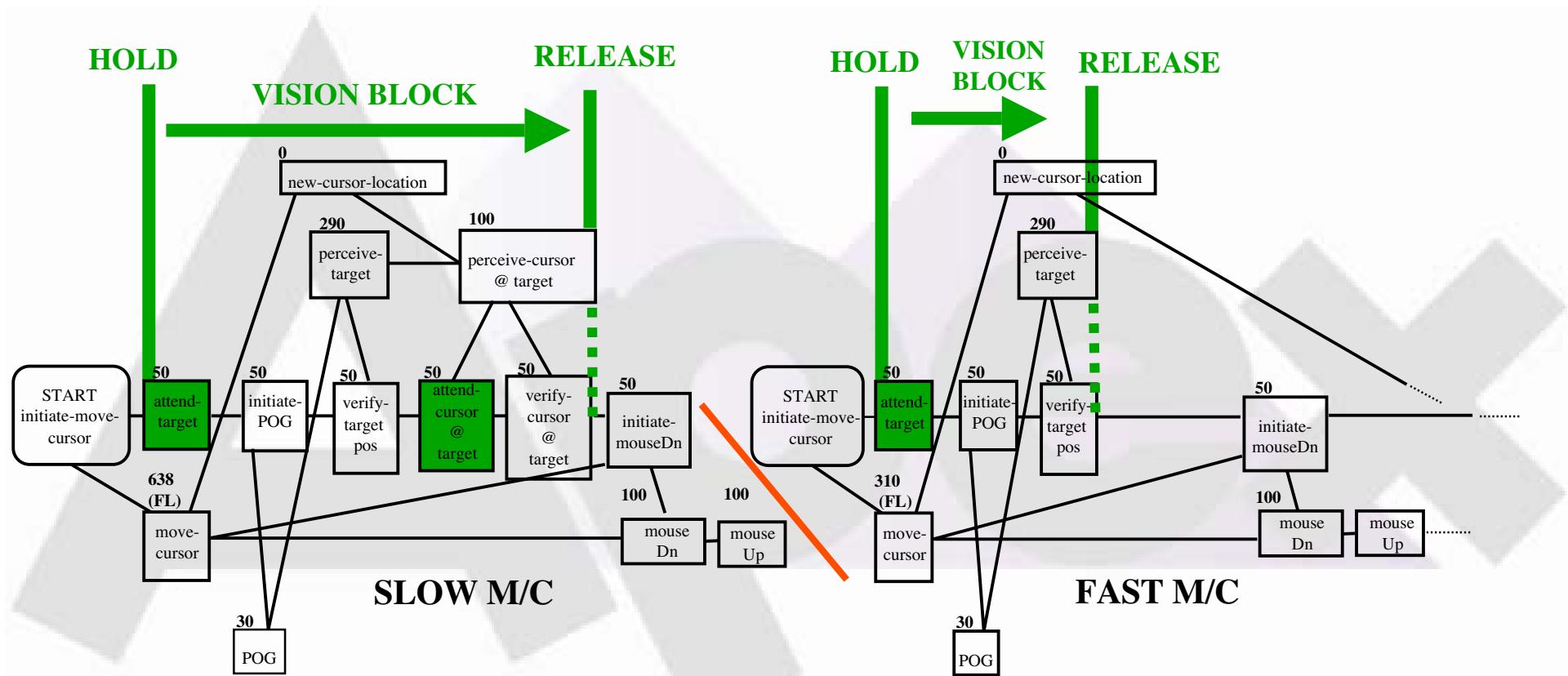


1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



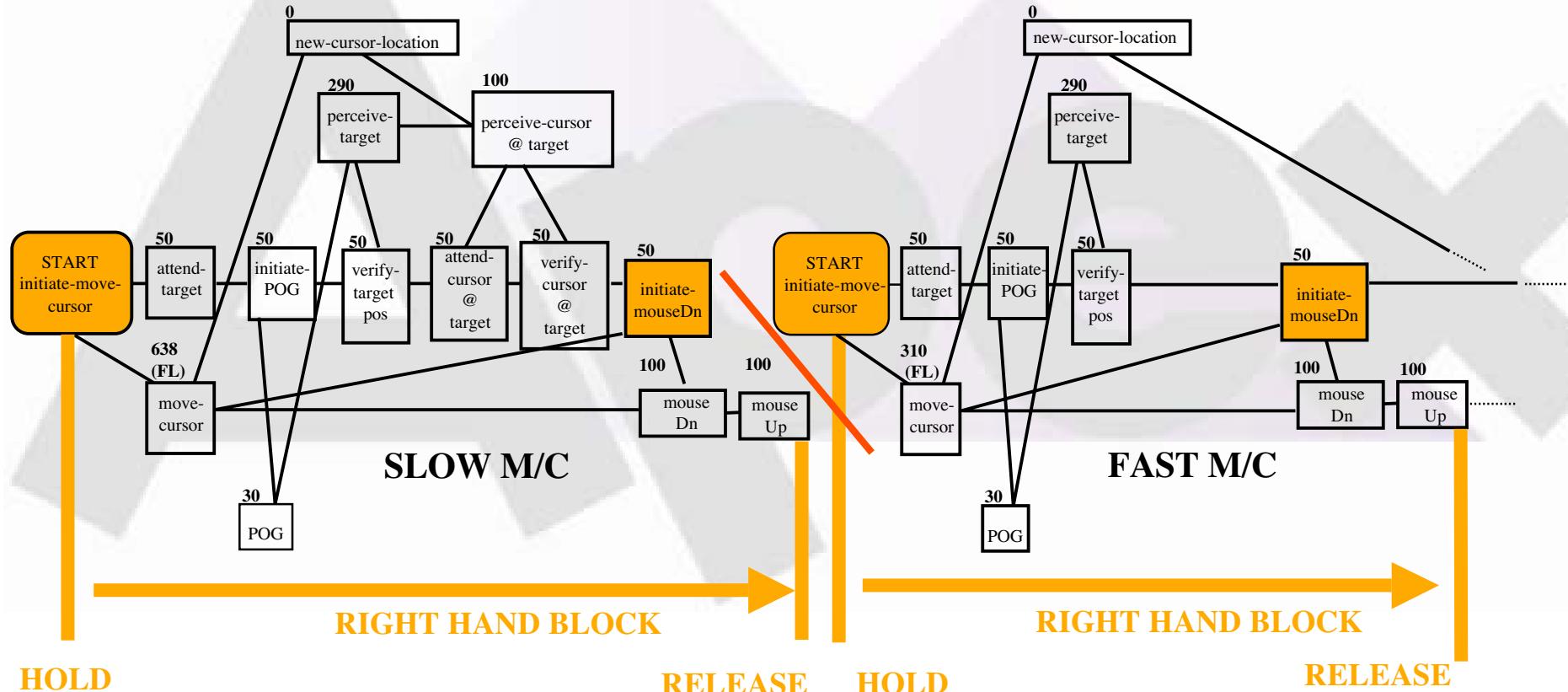
Interleaving Templates w/ Apex



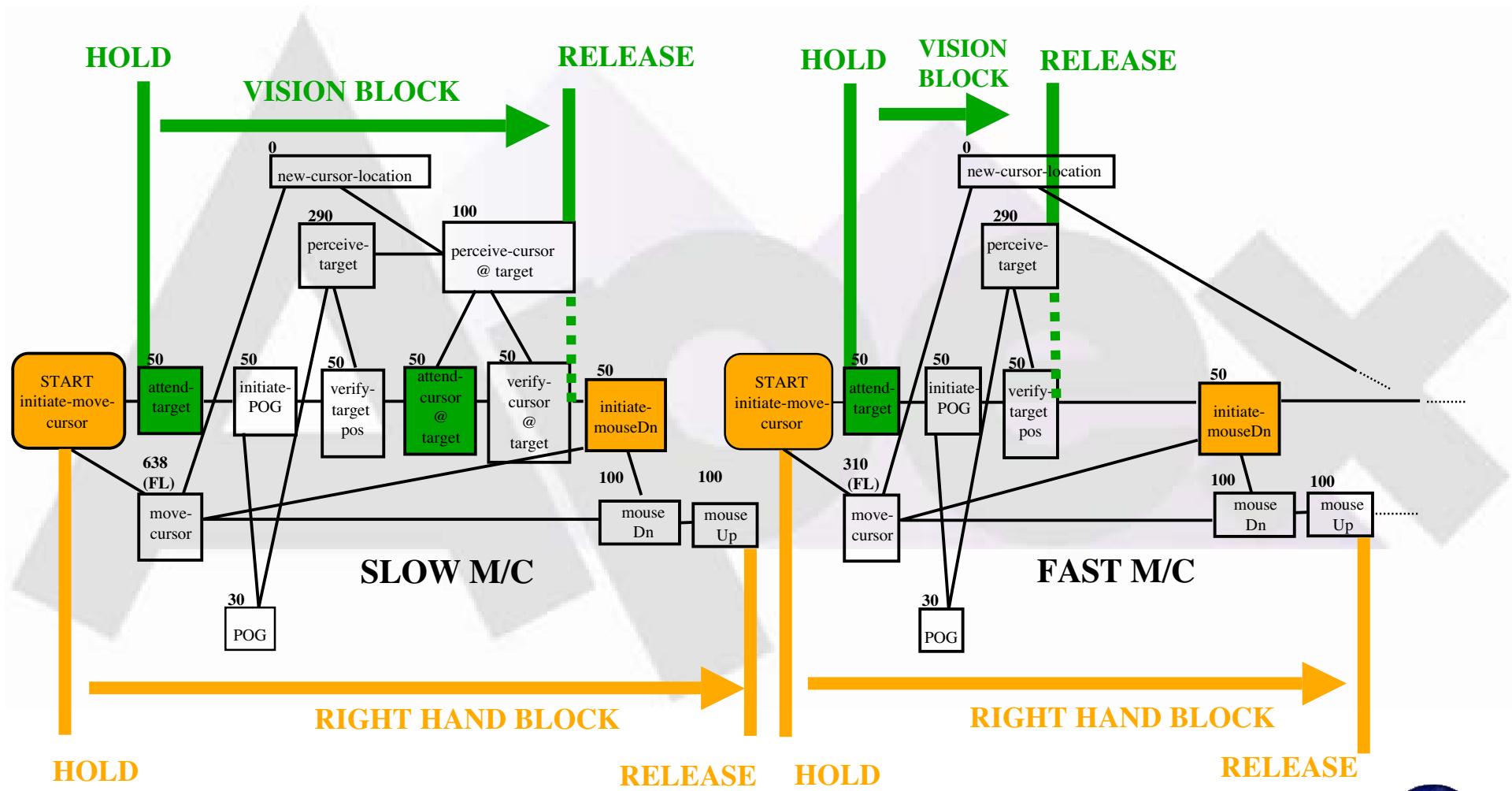
-preventing vision from getting ahead of itself

Interleaving Templates w/ Apex

-preventing the right hand from getting ahead of itself



Interleaving Templates w/ Apex

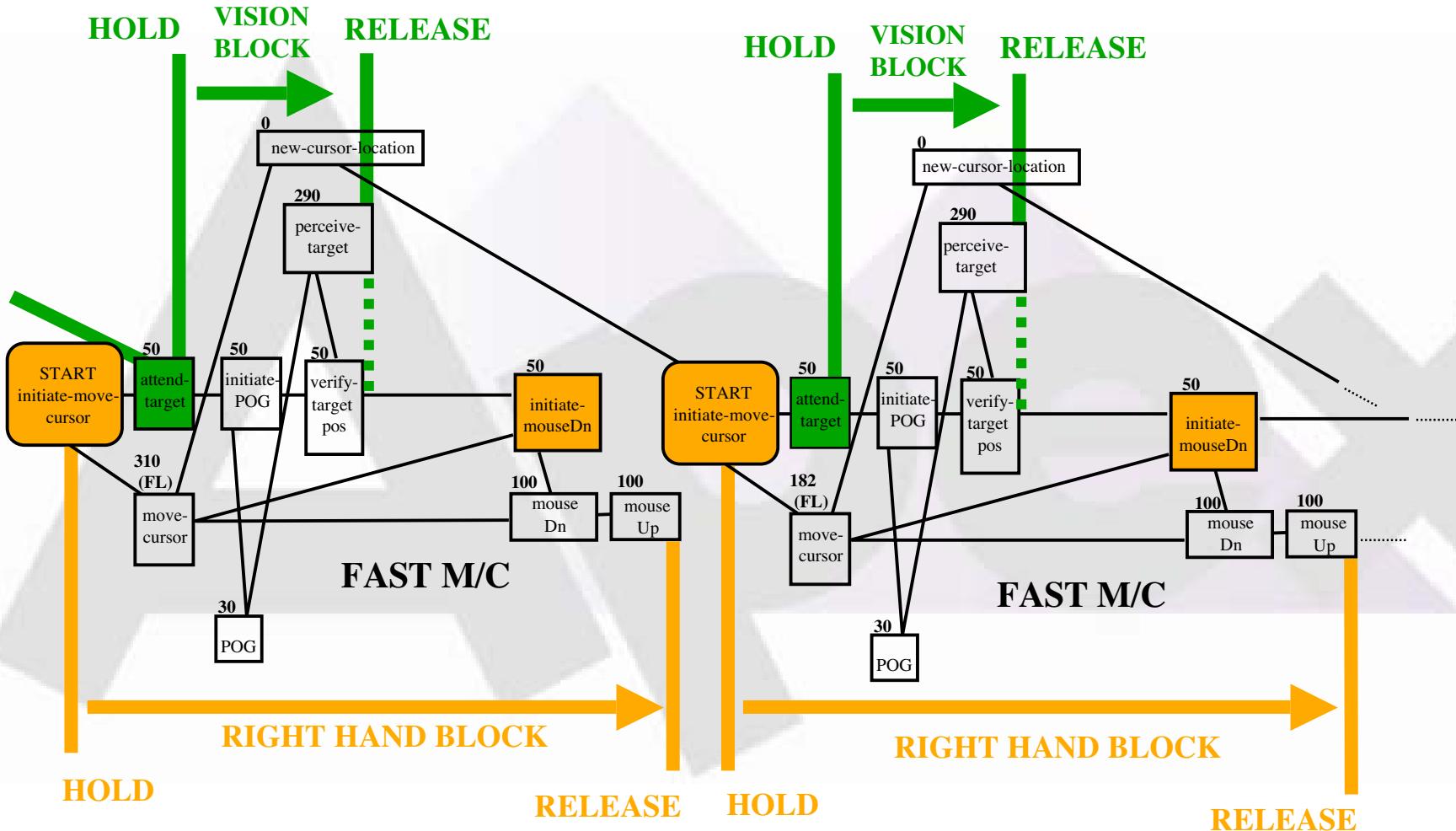


1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



Interleaving Templates w/ Apex

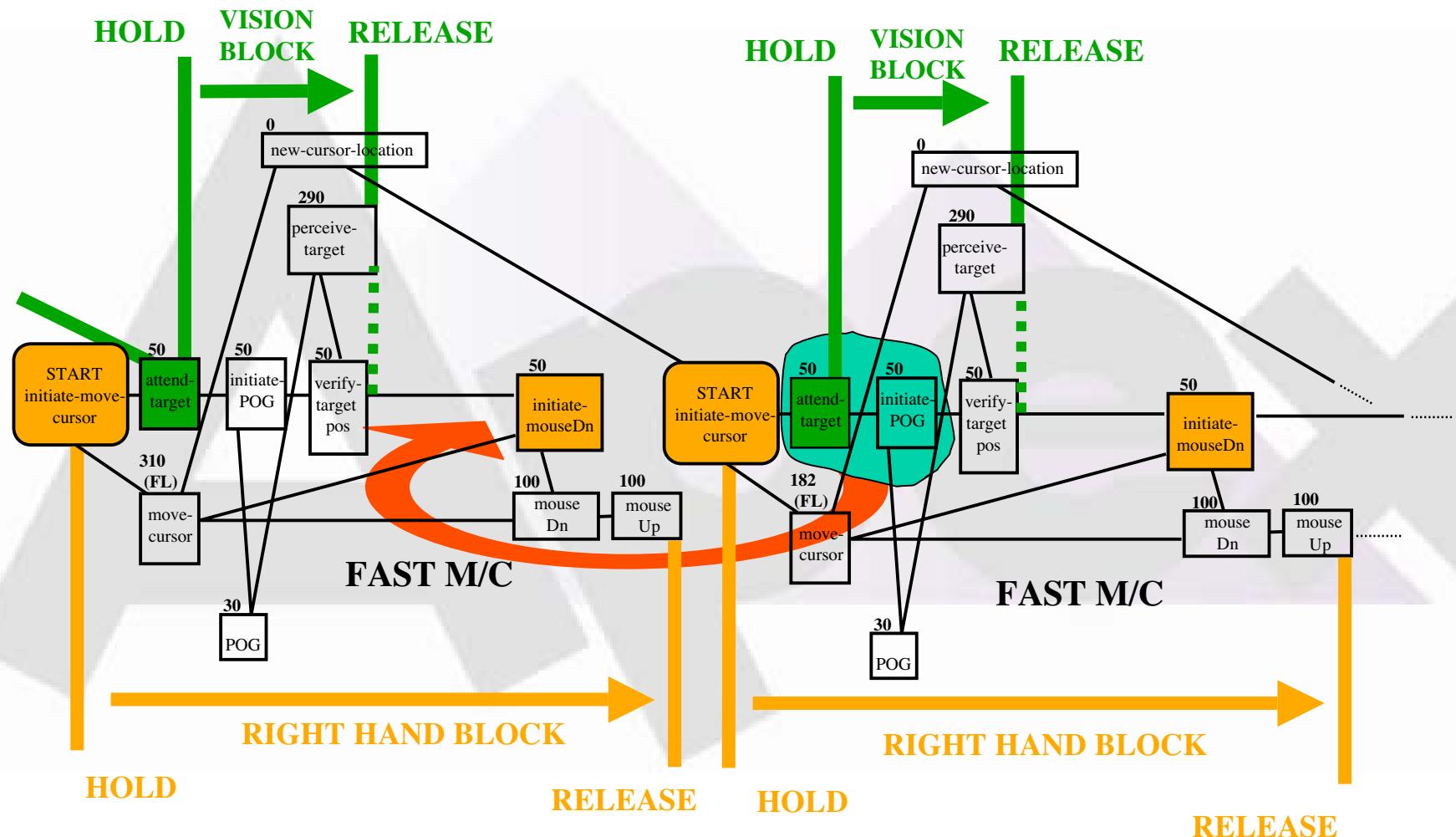


1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



Interleaving Templates w/ Apex

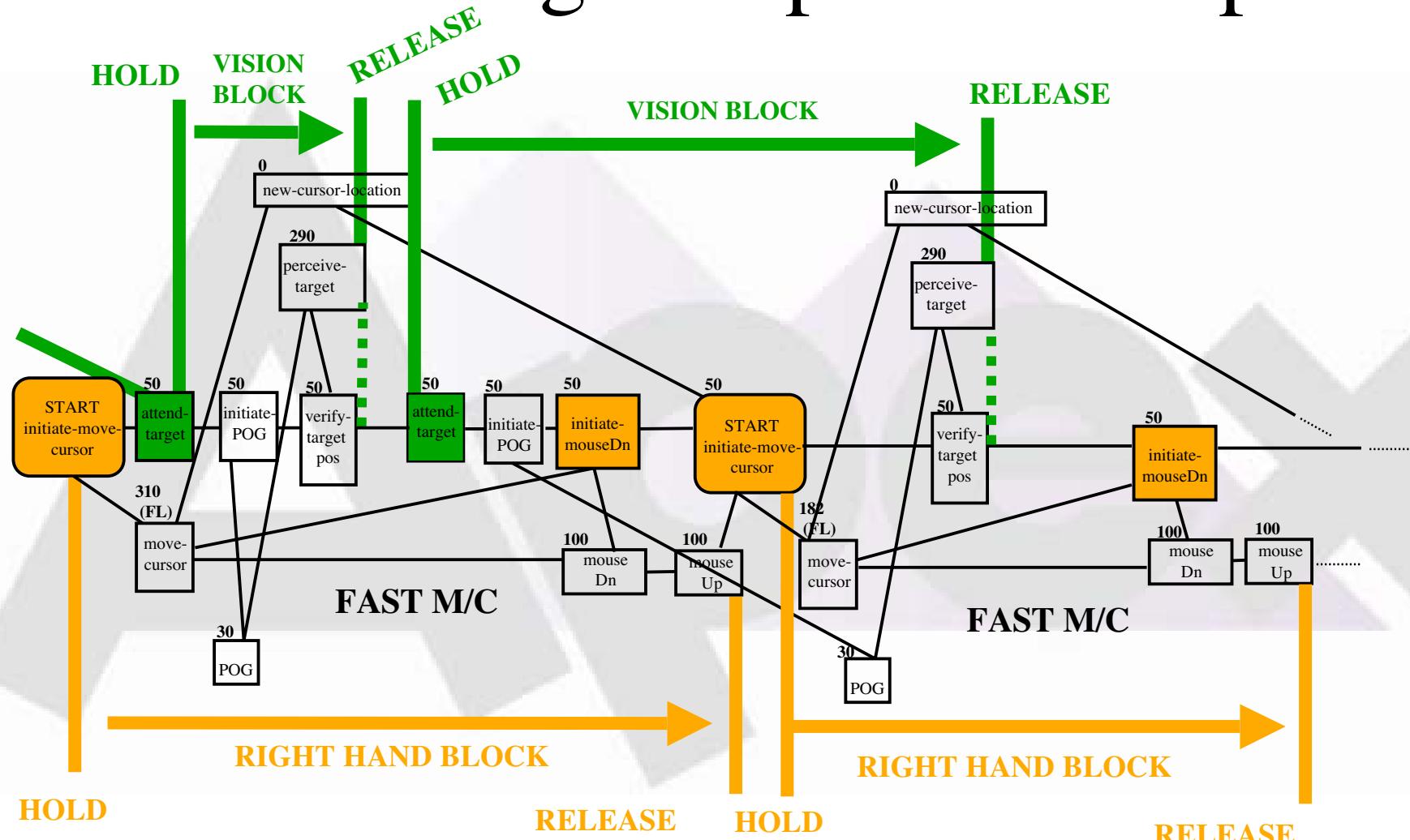


1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



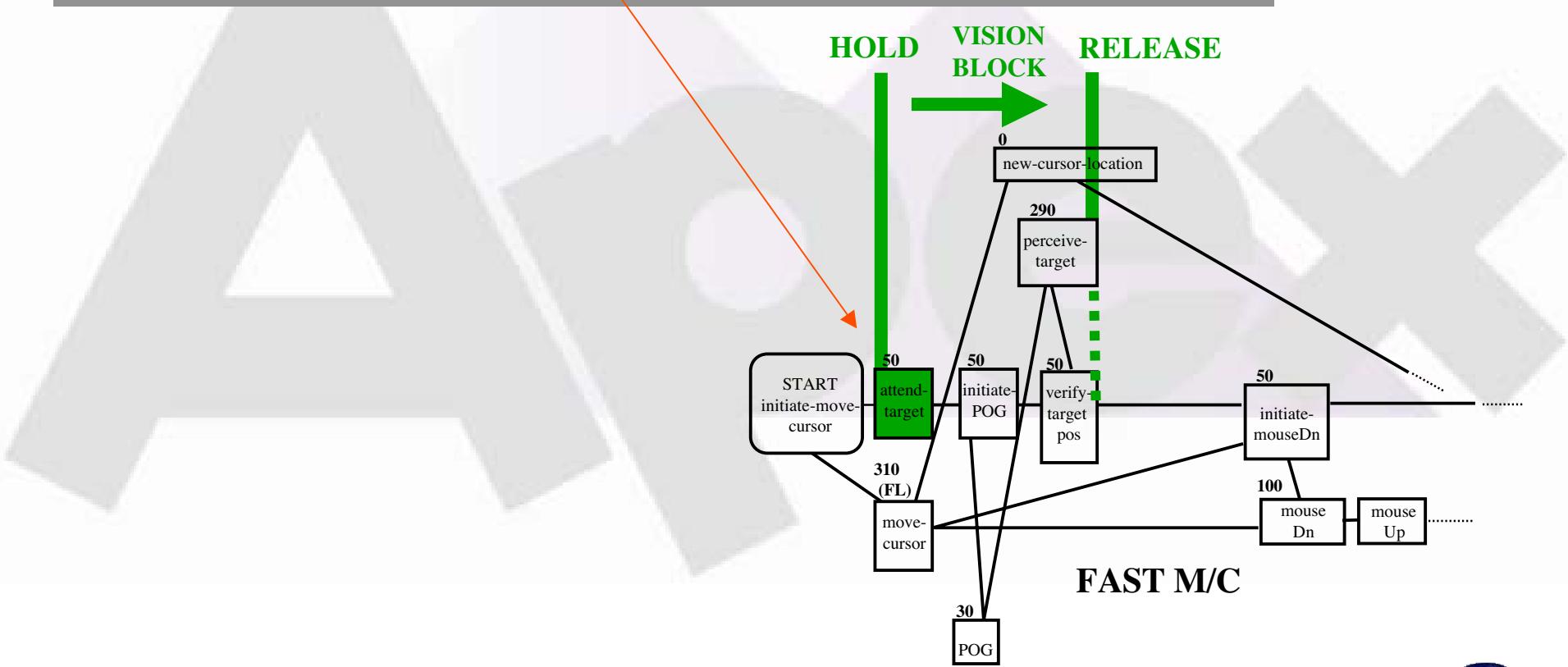
Interleaving Templates w/ Apex



```

(procedure
  (index (attend-target ?target))
  (profile memory vision-block)
  (step s1 (start-activity memory memory-act :duration 50 => ?a))
  (step s2 (hold-resource vision-block :ancestor 2)
    (waitfor (completed ?a)))
  (step r (reset ?self) (waitfor (resumed ?self)))
  (step s3 (terminate) (waitfor ?s2)))

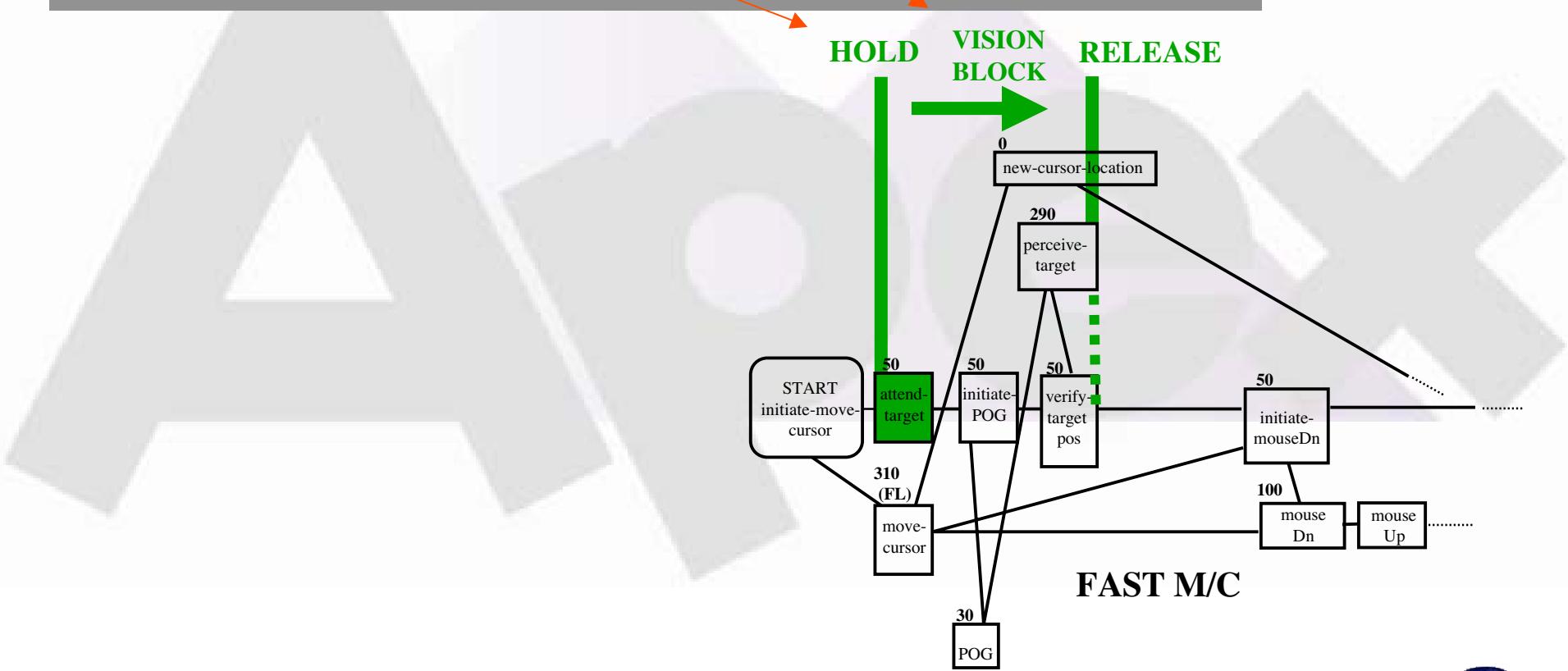
```



```

(procedure
  (index (attend-target ?target))
  (profile memory vision-block)
  (step s1 (start-activity memory memory-act :duration 50 => ?a))
  (step s2 (hold-resource vision-block :ancestor 2)
    (waitfor (completed ?a)))
  (step r (reset ?self) (waitfor (resumed ?self)))
  (step s3 (terminate) (waitfor ?s2)))

```



```
(procedure
  (index (attend-target ?target))
  (profile memory vision-block)
  (step s1 (start-activity memory memory-act :duration 50 => ?a))
  (step s2 (hold-resource vision-block :ancestor 2)
    (waitfor (completed ?a)))
  (step r (reset ?self) (waitfor (resumed ?self)))
  (step s3 (terminate) (waitfor ?s2)))
```

What happens when an operator doesn't fit completely in a space where there is slack time?

It needs to be interrupted without completing and restarted in the next slot.

Rule:

every procedure that uses resources (including anything with a start-activity step and anything that might be the target of a hold-resource) should have an interruption-handling behavior.

The above step form is a good default interrupt-handler



CPM-GOMS PDL

```
(procedure
  (index (do banking))
  (step s1 (initiate session))
  (step s2 (do transaction) (waitfor ?s1))
  (step s3 (end session) (waitfor ?s2))
  (step s4 (terminate) (waitfor ?s3)))
```

add
priorities

```
(procedure
  (index (do banking))
  (step s1 (initiate session) (priority ?))
  (step s2 (do transaction) (priority ?))
  (step s3 (end session) (priority ?))
  (step s4 (terminate) (waitfor ?s3 ?s2 ?s1)))
```

CPM-GOMS PDL

```
(procedure  
  (index (choose withdraw))  
  (step s2 (terminate) (waitfor ?s1)))
```

use
templates

```
(procedure  
  (index (choose withdraw))  
  (step s1 (?-move-click ?))  
  (step s2 (terminate) (waitfor ?s1)))
```



CPM-GOMS PDL

```
(procedure
  (index (do banking))
  (step s1 (initiate session))
  (step s2 (do transaction) (waitfor ?s1))
  (step s3 (end session) (waitfor ?s2))
  (step s4 (terminate) (waitfor ?s3)))
```

add
priorities

```
(procedure
  (index (do banking))
  (step s1 (initiate session) (priority ?))
  (step s2 (do transaction) (priority ?))
  (step s3 (end session) (priority ?))
  (step s4 (terminate) (waitfor ?s3 ?s2 ?s1)))
```

CPM-GOMS PDL

```
(procedure
  (index (do banking))
  (step s1 (initiate session))
  (step s2 (do transaction) (waitfor ?s1))
  (step s3 (end session) (waitfor ?s2))
  (step s4 (terminate) (waitfor ?s3)))
```

add
priorities

```
(procedure
  (index (do banking))
  (step s1 (initiate session) (priority 300))
  (step s2 (do transaction) (priority 200))
  (step s3 (end session) (priority 100))
  (step s4 (terminate) (waitfor ?s3 ?s2 ?s1)))
```

CPM-GOMS PDL

```
(procedure  
  (index (choose withdraw))  
  (step s2 (terminate) (waitfor ?s1)))
```

use
templates

```
(procedure  
  (index (choose withdraw))  
  (step s1 (?-move-click ?))  
  (step s2 (terminate) (waitfor ?s1)))
```



CPM-GOMS PDL

```
(procedure  
  (index (choose withdraw))  
  (step s2 (terminate) (waitfor ?s1)))
```

use
templates

```
(procedure  
  (index (choose withdraw))  
  (step s1 (fast-move-click withdraw-key))  
  (step s2 (terminate) (waitfor ?s1)))
```



Calculator World Exercise

load simworld CALC-YOURS

1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



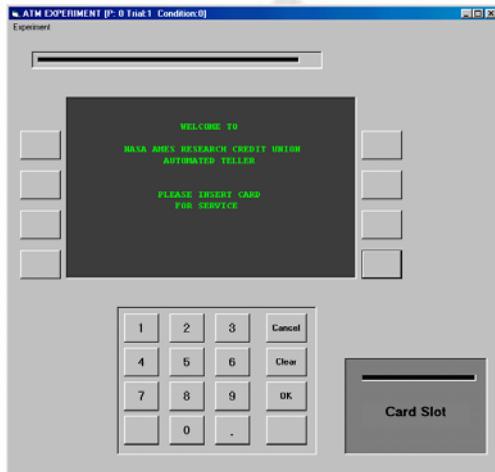
Calculator World



Task:

- 1) create, modify and delete keys**
- 2) write PDL to do
“2+2=“**

Calculator World



keys 4, 5, & 6
don't change

Position is measured
in *mm* from upper left
corner which is (0 0)

	Position	Dimensions
SCREEN	(43 90)	(62 11)
7-KEY	(43 104)	(13 11)
8-KEY	(60 104)	(13 11)
9-KEY	(77 104)	(13 11)
MINUS-KEY	(95 104)	(13 11)
PLUS-KEY	(95 117)	(13 11)
1-KEY	(43 130)	(13 11)
2-KEY	(60 130)	(13 11)
3-KEY	(77 130)	(13 11)
EQUAL-KEY	(95 129)	(13 22)
0-KEY	(43 143)	(26 11)
PERIOD-KEY	(77 143)	(13 11)



Calculator World

Making Interface Objects

```
(initialize-simulation
  (let*
    ((world (make-instance 'locale :name 'world))
     (human (make-instance 'human :name 'agent :locale world
                           :location '(0 0 400)))
     (calc (make-instance 'interface-object :name 'calc :locale world ))
     ; (money-slot (make-instance 'money-slot :name 'money-slot :locale world))
     (screen (make-instance 'screen :name 'screen :locale world))
     (minus-key (make-instance 'button :name 'minus-key :locale world))
     ; (withdraw-key (make-instance 'button :name 'withdraw-key :locale world))
     (period-key (make-instance 'button :name 'period-key :locale world))
     (plus-key (make-instance 'button :name 'plus-key :locale world))
     (keypad (create-keypad '(43 104) '(13 11) '(4 2) 'down world))
     (equal-key (make-instance 'button :name 'equal-key :locale world))
     ; (card-slot (make-instance 'card-slot :name 'card-slot :locale world))
     (mouse (create-mouse world))
     )
    (mapc 'assemble (list human calc ; card-slot money-slot withdraw-key
                           screen keypad
                           minus-key period-key mouse equal-key plus-key)))
```



Calculator World

Making Interface Objects

```
; Calculator screen
(set-visual-vals
  '(
    (0-KEY :pos (43 143) :dim (26 11))
    (7-KEY :pos (43 104) :dim (13 11))
    (8-KEY :pos (60 104) :dim (13 11))
    (9-KEY :pos (77 104) :dim (13 11))
    (4-KEY :pos (43 117) :dim (13 11))
    (5-KEY :pos (60 117) :dim (13 11))
    (6-KEY :pos (77 117) :dim (13 11))
    (1-KEY :pos (43 130) :dim (13 11))
    (2-KEY :pos (60 130) :dim (13 11))
    (3-KEY :pos (77 130) :dim (13 11))
    (CALC          :pos (0 0) :dim (182 164))
    (SCREEN         :pos (43 90) :dim (62 11))
    (KEYPAD         :pos (43 104) :dim (47 50))
;     (CARD-SLOT      :pos (131 127) :dim (43 2))
;     (MONEY-SLOT     :pos (11 9) :dim (98 2))
    (minus-KEY      :pos (95 104) :dim (13 11))
;     (WITHDRAW-KEY   :pos (0 0) :dim (0 0))
    (period-KEY     :pos (77 143) :dim (13 11))
    (plus-KEY       :pos (95 117) :dim (13 11))
    (equal-KEY      :pos (95 129) :dim (13 22))
  )))
))
```



Calculator World

PDL

```
(procedure  
  (index (do banking))  
    (step s1 (initiate session) (priority 300))  
    (step s2 (do transaction) (priority 200))  
    (step s3 (end session) (priority 100))  
    (step s4 (terminate) (waitfor ?s3 ?s2 ?s1)))
```

change
goals/steps

```
(procedure  
  (index (do-domain))  
    (step s1 (?-move-click 2-key) (priority ?))  
    (step s2 (?-move-click plus-key) (priority ?))  
    (step s3 (?-move-click 2-key) (priority ?))  
    (step s4 (?-move-click equal-key) (priority ?))  
    (step s5 (stop) (waitfor ?s1 ?s2 ?s3 ?s4)))
```



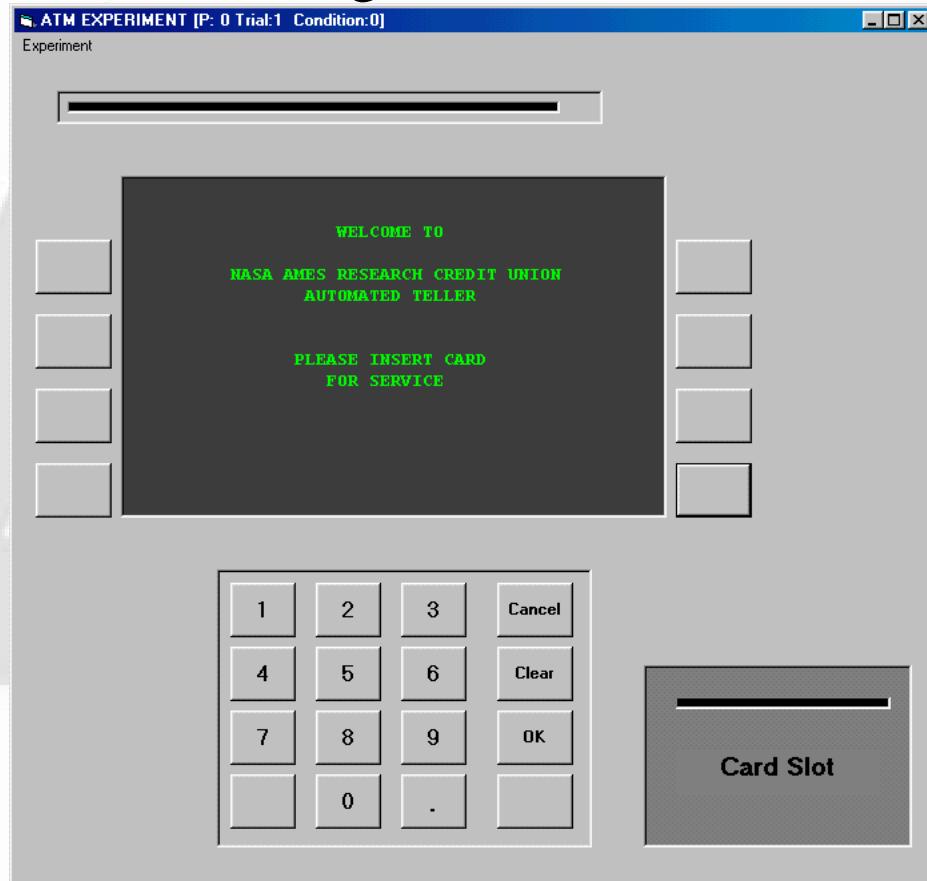
Text Legibility Example

1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



Layout #1: lime (0 255 0) on (46 46 46); letter size 0.28°
RT of reading: 3349 ms



Layout #2: green (0 128 0) on (46 46 46); letter size 0.28°
RT of reading: 3463 ms



Layout #3: green (0 128 0) on (46 46 46); letter size 0.21°
RT of reading: 3977 ms



Wrap-Up

1 August 2001

Cognitive Science 2001
Edinburgh, Scotland



What we did get to show is

- A modeling environment with tools to
 - Inspect models and output
 - Trace model activity
 - Analyze and represent model output (pert charts)
- Examples of building blocks for
 - Reusable behavior templates
 - Agent resources
 - World widgets
- A fast, automatic, consistent integration of model elements that yields improved prediction
- A significant enhancement in modeling capability



What we did not get to today

- Multiple task management
- Error prediction
- Automated Design Walkthrough
- World modeling
- Distributed community of users and developers

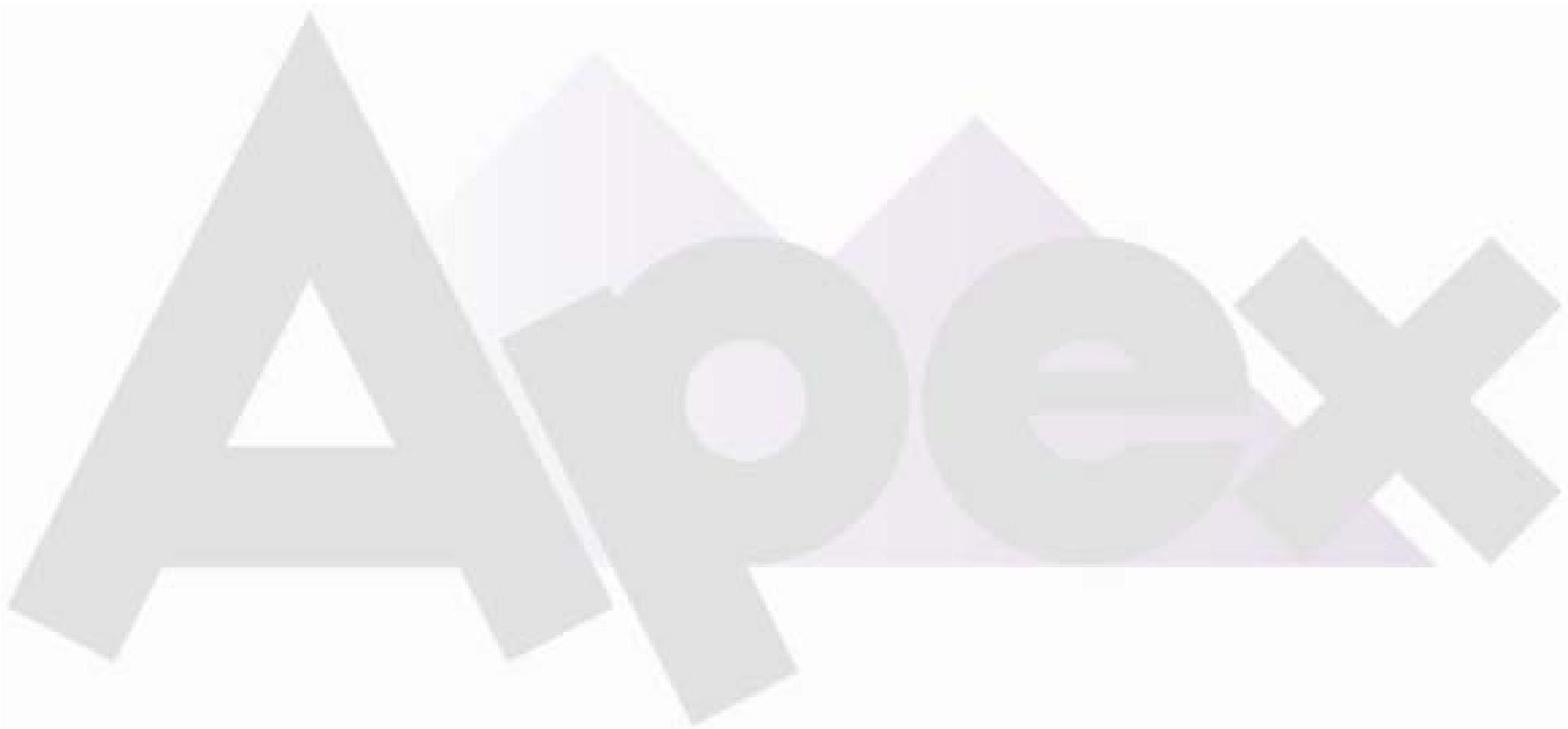
VISION

Faster
Easier
Fewer mistakes



Large-Scale
Models

Haggis, Neeps and Tatties



1 August 2001

Cognitive Science 2001
Edinburgh, Scotland

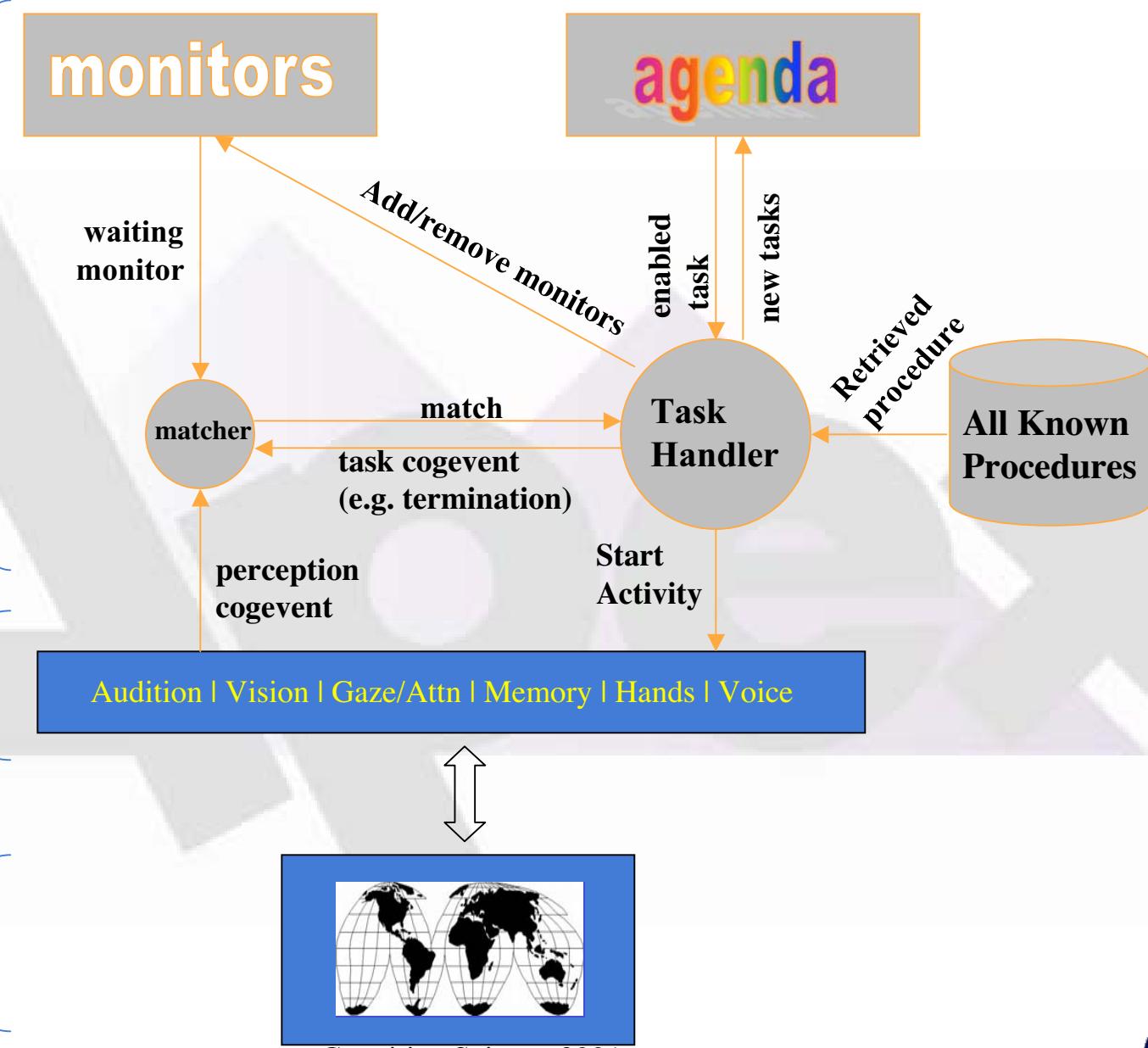


Action
Selection
Architecture

Human
Resource
Architecture

The World

1 August 2001



Cognitive Science 2001
Edinburgh, Scotland

